National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

UNIVERSITY OF ALBERTA

THE COMPUTER-CONTROLLED HEAT PUMP SYSTEM

By

KIM VU

(C)

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

DEPARTMENT OF ELECTRICAL ENGINEERING

Edmonton, Alberta

Fall 1991

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-69923-X

Canada

UNIVERSITY OF ALBERTA
RELEASE FORM

NAME OF AUTHOR: Kim Vu

TITLE OF THESIS: The Computer-Controlled Heat Pump System.

DEGREE: Master of Science in Electrical Engineering.

YEAR THIS DEGREE GRANTED: Fall 1991.

Permission is hereby granted to the University of Alberta library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Kim Vu

12724-90 Street
Edmonton, Alberta
Canada. T5E 3L6

September 5, 1991.

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *The Computer-Controlled Heat Pump System* submitted by Kim Vu in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

R. E. Rink

Supervisor: Dr. R.E. Rink

V. G. Gourishankar

Dr. V.G. Gourishankar

S. Shah

Dr. S. Shah

September 5, 1991.

Kính dâng cha mẹ
và
thương tặng Đỗ Thị Trâm.

ABSTRACT

A computer-based instrumentation and control system is implemented as a tool to study and optimize the performance of a water -to-water heat pump in space heating applications. The system is centered around a personal computer which performs such versatile tasks as data acquisition, process monitoring and control. Various sensors to measure temperature, pressure and flow etc. are interfaced to the computer where analog to digital conversions are performed by a commercially-available board installed inside the computer. Two motorized valves are designed and under digital proportional-integral-derivative control, whereby the effects of the source temperature and the heating load on the performance of the heat pump can be studied. In addition, a computer program is developed for real time process control, and it provides extensive operator-plant interface through the keyboard and screen display.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. R.E. Rink, my supervisor, for setting the direction of my research, guidance, unfailing patience and financial support in the course of my study. I am also grateful to Dr. V.G. Gourishankar who has facilitated my research by allowing me easy access to computer facilities, to Dr. J. King and Mr. Dang Trung for valuable technical discussions, and finally to Mr. Alex Srinivasan for drawing the circuit diagrams in this thesis.

CONTENTS

	Page
Chapter 1: INTRODUCTION	1
1.1 Background	1
1.2 Principle of operation	2
1.3 Performance optimization	4
1.3.1 Superheat control	5
1.3.2 Capacity control	6
1.4 Research objectives	6
1.5 Thesis organization	8
Figures	10
Chapter 2: INSTRUMENTATION	11
2.1 The heat pump	11
2.2 Measuring sensors and transducers	14
2.2.1 Temperature measurement	15
2.2.2 Pressure measurement	16
2.2.3 Flow measurement	16
2.2.4 Power measurement	17
2.2.5 On/Off switching of the compressor	17
2.2.6 Final control element: The motorized valve	17
2.3 Calibration	19
2.4 The data acquisition system (DAS)	19
2.5 The computer IBM PC/XT	21
Figures	23

Table ---- 31

Chapter 3: SOFTWARE DEVELOPMENT ---- 32
3.1 Program structure ---- 32
3.2 System memory ---- 34
3.2.1 Accessing system memory for graphic display ---- 34
3.3 Interrupts ---- 37
3.3.1 The 8259A Programmable Interrupt Controller (PIC) 37
3.3.2 Setting up the interrupt vector for IRQ2 interrupt handler ---- 41
3.3.3 Servicing DAS interrupts ---- 42
3.3.4 The *handler ()* routine ---- 44
3.4 The data acquisition system (DAS) ---- 45
3.4.1 Programming the System Timing Controller (STC) 45
3.4.2 A/D Conversion ---- 48
3.4.3 Programming the Peripheral Interface chip (PI) ---- 50
3.5 PID screen ---- 50
3.6 Display screen ---- 52
3.6.1 Starting the DAS ---- 52
3.6.2 Stopping The DAS ---- 53
3.7 Help screen ---- 54
3.8 Evaluation of refrigerant thermodynamic properties ---- 54
Figures ---- 57
Tables ---- 60
Flow charts ---- 66

Chapter 4: CONTROLLER TUNING ---- 72
4.1 Controlling the water flow rate into the condenser (F_o) ---- 73
4.1.1 Tuning the controller ---- 73
4.1.2 Experimental results ---- 75
4.2 Controlling the evaporator inlet water temperature (T_o) ---- 76
4.2.1 Dead time compensatiom ---- 76
4.2.2 Describing function method ---- 80
4.2.3 Tuning the temperature controller ---- 84
4.2.4 Experimental results ---- 85
Figures ---- 87
Tables ---- 89
Figures ---- 91

Chapter 5: CONCLUSION ---- 117
REFERENCES ---- 119

LIST OF TABLES

		Page
Table 1:	Calibration.	31
Table 2:	Keyboard assignment.	60
Table 3:	Cursor control sequences.	61
Table 4:	8088 interrupt priorities.	61
Table 5:	8088 hardware interrupt listing.	62
Table 6:	Addressing structure of the PIC.	62
Table 7:	Frequency sources for DAS counter 1.	62
Table 8:	Input/Output mapping of the A/D converter.	63
Table 9:	Flags associated with PID screen.	63
Table 10:	Flags associated with Display screen.	64
Table 11:	Coefficients to evaluate refrigerant thermodynamic properties.	65
Table 12:	Controller's settings for the flow control loop.	89
Table 13:	Process characteristics of the temperature control loop.	89
Table 14:	Effects of modeling errors on process stability.	90
Table 15:	Controller's settings for the temperature control loop.	90

LIST OF FLOW CHARTS

		Page
Flow chart 1:	PIC initialization.	66
Flow chart 2a:	*handler()* routine, Part A.	67
Flow chart 2b:	*handler()* routine, Part B.	68
Flow chart 3a:	*starter()* routine.	69
Flow chart 3b:	*stopper()* routine.	69
Flow chart 4:	*measurement_display()* routine.	70
Flow chart 5:	Help screen.	71

LIST OF FIGURES

		Page
Fig.1.1a:	Basic heat pump components.	10
Fig.1.1b:	Ideal vapor-compression cycle.	10
Fig.1.1c:	Heat pump working cycle.	10
Fig.2.1:	Heat pump block diagram.	23
Fig.2.2a:	Three-wire bridge.	24
Fig.2.2b&c:	Temperature measurement.	24
Fig.2.3a:	Low pressure measurement.	25
Fig.2.3b:	High pressure measurement.	25
Fig.2.4:	Source water flow measurement.	25
Fig.2.5:	Cold water flow measurement.	26
Fig.2.6:	Power sensor.	27
Fig.2.7:	Compressor switching circuitry.	27
Fig.2.8:	Circuit diagram of motorized valve 1.	28
Fig.2.9:	Circuit diagram of motorized valve 2.	29
Fig.2.10:	Block diagram of the computer-controlled heat pump system.	30
Fig.3.1:	Program structure.	57
Fig.3.2:	Block diagram of menu-driven program.	57
Fig.3.3:	Video memory to screen mapping.	58
Fig.3.4:	A far pointer to video memory.	58
Fig.3.5:	8088 interrupt sources.	58
Fig.3.6:	Relationship between T_{s1} and T_{s2}.	59
Fig.3.7:	DAS interrupt source.	59

Fig.3.8:	Arrangement of control signal to motorized valves	59
Fig.3.9:	Block diagram of flow control loop.	59
Fig.3.10:	Block diagram of compensated temperature loop.	59
Fig.4:	Relay characteristics.	87
Fig.4.1a:	Flow control loop.	87
Fig.4.2a:	Temperature control loop.	87
Fig.4.2b:	A typical process reaction curve	88
Fig.4.3:	Block diagram of a nonlinear feedback system	88

LIST OF RESULTS

		Page
FLOW CONTROL		
Experimental results		
Fig.4.1.1:	PID control (F_{sp} = 16L/Min).	91
Fig.4.1.2:	PID control (F_{sp} = 7L/Min).	91
Fig.4.1.3:	PID control (F_{sp} = 4L/Min).	92
Fig.4.1.4:	PID control (F_{sp} = 20L/Min).	92
Fig.4.1.5:	PID control (F_{sp} = 10L/Min).	93
Fig.4.1.6:	PID control (F_{sp} = 13L/Min).	93
Fig.4.1.7:	Effect of disturbance (F_{sp} = 18L/Min).	94
Fig.4.1.8:	Effect of disturbance (F_{sp} = 5L/Min).	94
Fig.4.1.9:	Effect of disturbance (F_{sp} = 14L/Min).	95
Fig.4.1.10:	Effect of disturbance (F_{sp} = 12L/Min).	95
TEMPERATURE CONTROL		
a. Simulation		
Fig.4.3.1a:	Nyquist plot of $G_p(s)$.	96
Fig.4.3.1b:	Simulation result (T_{sp} = 16°C).	96
b. Modeling error consideration		
Fig.4.3.2a:	Nyquist plot of $G_p(s)$ + $G_m(s)$.	97
Fig.4.3.2b:	Simulation result (T_{sp} = 16°C).	97
Fig.4.3.3a:	Nyquist plot of $G_p(s)$ + $G_m(s)$.	98
Fig.4.3.3b:	Simulation result (T_{sp} = 16°C).	98
Fig.4.3.4a:	Nyquist plot of $G_p(s)$ + $G_m(s)$.	99
Fig.4.3.4b:	Simulation result (T_{sp} = 16°C).	99

Fig.4.3.5a:	Nyquist plot of $G_p(s) + G_m(s)$.	100
Fig.4.3.5b:	Simulation result (T_{sp} = 16°C).	100
Fig.4.3.6a:	Nyquist plot of $G_p(s) + G_m(s)$.	101
Fig.4.3.6b:	Simulation result (T_{sp} = 16°C).	101
Fig.4.3.7a:	Nyquist plot of $G_p(s) + G_m(s)$.	102
Fig.4.3.7b:	Simulation result (T_{sp} = 16°C).	102
Fig.4.3.8a:	Nyquist plot of $G_p(s) + G_m(s)$.	103
Fig.4.3.8b:	Simulation result (T_{sp} = 16°C).	103
Fig.4.3.9a:	Nyquist plot of $G_p(s) + G_m(s)$.	104
Fig.4.3.9b:	Simulation result (T_{sp} = 16°C).	104
Fig.4.3.10a:	Nyquist plot of $G_p(s) + G_m(s)$.	105
Fig.4.3.10b:	Simulation result (T_{sp} = 16°C).	105
Fig.4.3.11a:	Nyquist plot of $G_p(s) + G_m(s)$.	106
Fig.4.3.11b:	Simulation result (T_{sp} = 16°C).	106
c. Tuning the controller		
Fig.4.4.1:	Effect of P mode.	107
Fig.4.4.2:	Effect of I mode.	107
Fig.4.4.3:	Effect of PD mode.	108
Fig.4.4.4:	Effect of PID mode.	108
d. Experimental results		
Fig.4.5.1:	P control (T_{sp} = 14°C).	109
Fig.4.5.2:	P control (T_{sp} = 18°C).	109
Fig.4.5.3:	P control (T_{sp} = 18°C).	110
Fig.4.5.4:	P control (T_{sp} = 20°C).	110
Fig.4.5.5:	P control (T_{sp} = 18°C).	111
Fig.4.5.6:	PID control (T_{sp} = 14°C).	111

Fig.4.5.7:	PID control (T_{sp} = 16°C).	112
Fig.4.5.8:	PID control (T_{sp} = 18°C).	112
Fig.4.5.9:	PID control (T_{sp} = 14°C).	113
Fig.4.5.10:	Effect of disturbance (T_{sp} = 20°C).	113
Fig.4.5.11:	Effect of disturbance (T_{sp} = 16°C).	114
Fig.4.5.12:	Effect of disturbance (T_{sp} = 14°C).	114
Fig.4.5.13:	Effect of disturbance (T_{sp} = 16°C).	115
Fig.4.5.14:	Effect of disturbance (T_{sp} = 18°C).	115
Fig.4.5.15:	Effect of disturbance (T_{sp} = 18°C).	116
Fig.4.5.16:	Effect of disturbance (T_{sp} = 20°C).	116

NOMENCLATURE

m_r	[Kg/s]	Refrigerant mass flow rate.
h_i	[J/Kg]	Enthalpy at point i. $1 \leq i \leq 4$.
Q_H	[J/s]	Rate of heat transfer to the condenser.
Q_L	[J/s]	Rate of heat transfer to the evaporator.
Q_{Loss}	[J/s]	Rate of heat loss from the condenser.
W	[J/s]	Power input to the compressor.
COP		Coefficient of performance.
m_{we}	[Kg/s]	Mass flow rate of evaporator water inlet.
m_{wc}	[Kg/s]	Mass flow rate of cold water into condenser.
c_p	[J/(Kg°C)]	Specific heat of water.
Ti	[°C]	Temperature at point i. $1 \leq i \leq 8$.
To (T6)	[°C]	Evaporator water inlet temperature.
Fo (Wflow)	[L/Min]	Volume flow rate of cold water into condenser.
Eva_flow	[L/Min]	Volume flow rate of evaporator water inlet.
P_H	[psi]	Discharge pressure.
P_L	[psi]	Suction pressure.
ai		Coefficient to evaluate refrigerant thermodynamic properties (Table 11).
DAS		Data acquisition system
PIC		Programmable Interrupt Controller.
PI		Peripheral Interface chip.
STC		System Timing Controller.

CHAPTER 1
INTRODUCTION

The heat pump is thermodynamically identical to the refrigerator; however, their objectives contrast sharply with one another. While the refrigerator rejects heat to create cooling effects, the heat pump utilizes heat for heating purposes.

1.1 Background

The history of the heat pump can be traced back to Carnot's time when he described the theoretical concept of the heat pump in 1824. Lord Kelvin was the first to propose the heat pump, or 'heat multiplier' as he called it, in 1852 [1].

Interest in the heat pump has been spurred largely by a search for alternative sources of energy to alleviate current energy problems. The oil crisis in 1973 was a case in point. The heat pump fits well in with energy conservation because the source from which heat is extracted can be found in abundance, e.g. ambient air, ground water, waste disposal etc.

With technological advances, the heat pump has found applications in both industrial and residential settings [2,3]. In Canada, following the energy crisis in 1973, the Heat Pump Research and Development program was initiated and a review of the program was reported in 1986 [4].

1.2 Principle of operation

Coming in different shapes and sizes, the heat pump mainly operates on the vapor-compression cycle (Fig.1.1b), and consists of four basic components: an evaporator, a compressor, a condenser, and an expansion valve (Fig.1.1a).

Applying the laws of thermodynamics around the ideal cycle under steady state conditions [5], one obtains:

Path 4 $\rightarrow$ 1 (isobaric and isothermal evaporation):

$$m_r(h_1 - h_4) - Q_L = 0 \qquad (1.1a)$$

Path 1 $\rightarrow$ 2 (isentropic compression):

$$m_r(h_2 - h_1) - W = 0 \qquad (1.1b)$$

Path 2 $\rightarrow$ 3 (isobaric and isothermal condensation):

$$m_r(h_3 - h_2) + Q_H = 0 \qquad (1.1c)$$

Path 3 $\rightarrow$ 4 (isenthalpic expansion):

$$m_r(h_4 - h_3) = 0 \qquad (1.1d)$$

Combining the above equations yields

$$Q_H = Q_L + W \qquad (1.2)$$

m_r represents the mass flow rate of the refrigerant in [Kg/s]. h_i ($1 \leq i \leq 4$) is the enthalpy at point i in [J/Kg]. Q_L and Q_H are the rates of heat transfer from the source, and to the sink, respectively. W is the power input to the compressor. Q_L, Q_H and W are all expressed in [J/s].

The coefficient of performance (COP) for the heat pump is defined as

$$COP = \frac{Q_H}{W} = \frac{1}{1 - Q_L/Q_H} \leq \frac{1}{1 - T_L/T_H} = COP_{Carnot} \quad (1.3)$$

COP represents energy "gain" for the heat pump. It is the ratio of energy delivered to the condenser to electrical energy consumed by the compressor motor. In (1.3), T_L and T_H are the source and sink temperature, respectively, in [K]. Since the COP of the heat pump is always greater than 1 and can never exceed COP_{Carnot}, the latter is normally used as a fundamental guideline for all heat pump applications. From (1.3) it is seen that the COP increases with a decrease in the temperature difference between the source and the sink.

In reality, a practical vapor-compression cycle for the heat pump departs from the ideal cycle (Fig.1.1c). In this practical cycle:

- The compression is no longer isentropic. The entropy will increase during compression.
- Pressure drops will occur during condensation and evaporation.
- The expansion is not adiabatic.
- Some degree of superheat is maintained at the evaporator outlet to ensure that the refrigerant is completely vaporized before entering the compressor.
- A temperature difference between the source and the refrigerant (working fluid) is required to facilitate heat transfer in the evaporator.

With the source being water, the rate of heat transfer to the refrigerant at the evaporator is given by

$$Q_L = m_{we} c_p (T_6 - T_8) \qquad (1.4)$$

In (1.4) m_{we} is the mass flow rate of water through the evaporator in [Kg/s]. c_p is the specific heat of water in [J/(Kg°C)]. T_6 and T_8 are the evaporator inlet and outlet water temperature, respectively, in [°C].

The rate of heat removed from the condenser, or the heat load on the heat pump, in the steady state is

$$m_{wc} c_p (T_4 - T_3) = Q_H - Q_{Loss} \qquad (1.5)$$

Again, in (1.5) T_3 and m_{wc} represent the temperature and mass flow rate of the cold water inlet of the condenser, respectively. T_4 is the temperature of the warm water outlet of the condenser. Q_{Loss} is the rate of heat loss from the condenser to the surroundings. m_{wc} is expressed in [Kg/s], the temperature in [°C], and the energy in [J/s].

Though simple as the concept may sound, it is a complex task of transforming this principle into a feasible operating system. Extensive writings on the heat pump can be found in the literature. Understandings of its working cycle, equipment constraints as well as operational problems have been well established. Through technical improvements and innovations, the heat pump today is commercially more viable in terms of improved reliability and performance.

1.3 Performance optimization

From process control perspective, optimizing the performance of the heat pump in terms of increasing its COP is a multivariable

control problem where its parameters influence each other.

1.3.1 Superheat control

The superheat, defined as the difference between the temperature of the superheated vapor at the outlet of the evaporator and the corresponding saturation temperature, provides the compressor with a safety margin against the entry of liquid refrigerant. A 10°C superheat is normally selected as a compromise between two contradicting demands: maximizing the evaporating surface and hence the rate of refrigerant evaporation on the one hand, and increasing the safety margin on the other.

The thermostatic expansion valve plays a major role in the thermodynamic cycle of the heat pump, and its dynamic characteristics have been described by Higuchi et al [6]. As a proportional superheat controller, the valve regulates the flow rate of the refrigerant into the evaporator in order to maintain the desired superheat at the evaporator outlet. Unfortunately, its performance is sometimes unsatisfactory, leading to unstable conditions. As described by Danig [7], Broersen et al [8], and subsequently confirmed by Grudhle et al [9] through theoretical modeling, hunting occurs when the valve undergoes frequent on/off cycling. In this unstable operation, the evaporator now becomes overfed then underfed, causing the superheat to oscillate about its set point. To improve performance, Gruhler [10] replaces the thermostatic expansion valve with a motorized needle valve under PID control. With this new control scheme, superheat control is greatly improved, resulting in an increase in the

evaporator efficiency.

1.3.2 Capacity control

In space heating, the heat load varies according to weather conditions. Some form of supplementary heat is required when the heat load exceeds the heat pump capacity. On the other hand, when heat supplied by the heat pump is greater than demand, it is necessary to reduce capacity in order to improve the heat pump efficiency.

In the conventional on/off control method, the compressor goes through the on/off cycle in order to match the load. With frequent light loads, the compressor life might be shortened due to frequent on/off cycling. Other capacity control methods include hot gas by-pass, cylinder unloading, clearance volume control, and speed control [11]; however, according to Wilson et al [12], only speed control is feasible for heat pump applications because the method results in a reduction in power consumed by the compressor.

In the work by Wang et al [13], an air to water heat pump has been designed to operate with a variable speed compressor, a motorized expansion valve, and a variable air mass flow rate. Independent control algorithms for the valve, compressor speed and air mass flow rate are developed. The heat pump is capable of delivering a maximum heat output of 8 KW, which represents an approximate heating load of a 3-bedroom house.

1.4 Research objectives

The heat pump used in the current research is intended for space

heating. The compressor is a constant speed type, and the source from which heat is extracted is water. Another cold water loop is used to remove heat from the condenser. The research objectives are to lay the ground work for future heat pump study from a computer-based instrumentation and control system where such 'bang bang' optimum control strategies as proposed by Rink et al [14] can be implemented. In this respect, the scope of the research includes:

a. Instrumentation:

- The heat pump is interfaced to the IBM PC/XT. A data acquisition system (DAS) installed inside the computer performs A/D conversions of analog signals coming from measurement sensors located at various points of the heat pump, at sampling intervals specified by the user.
- The compressor can be started and stopped either manually from the keyboard or under computer control. The compressor is connected to digital I/O lines via a relay circuitry.
- Two motorized valves are designed and under digital PID control. They each are connected to digital I/O lines via a relay circuitry. The valves can also be opened or closed manually from the keyboard for tuning purposes. Detection of fully closed/opened conditions is performed by a current-sensing resistor and protective measures are carried out by software.

b. Software development:

The program is menu-driven and consists of 3 screens:

- PID screen: This is the screen where the user enters controller parameters for both source temperature and cold water flow loops.
- Display screen: It is the most important screen of all. The sampling rates are specified and the DAS can be enabled or disabled. Sampled data are stored on file for off-line analysis and also displayed for viewing. Automatic control of temperature and cold water flow loops are carried out when enabled. The status of the valves, the DAS, and the compressor is constantly displayed. In addition, the times at which the experiment starts and stops are also displayed.
- Help screen: This screen provides the user with an easy access to the operation manual stored on file. The user can scroll the manual up or down for viewing.

c. Controller tuning:

- Tuning is required for temperature and flow control loops. For the temperature control loop, the Smith predictor is implemented with software to eliminate the effect of dead time.

1.5 Thesis organization

The thesis consists of five chapters. Chapter 1 is the introductory chapter. Chapter 2 deals with equipment description and

instrumentation. Software development is found in chapter 3. Controller tuning and results are presented in chapter 4, and chapter 5 provides the conclusion with recommendations for future work.

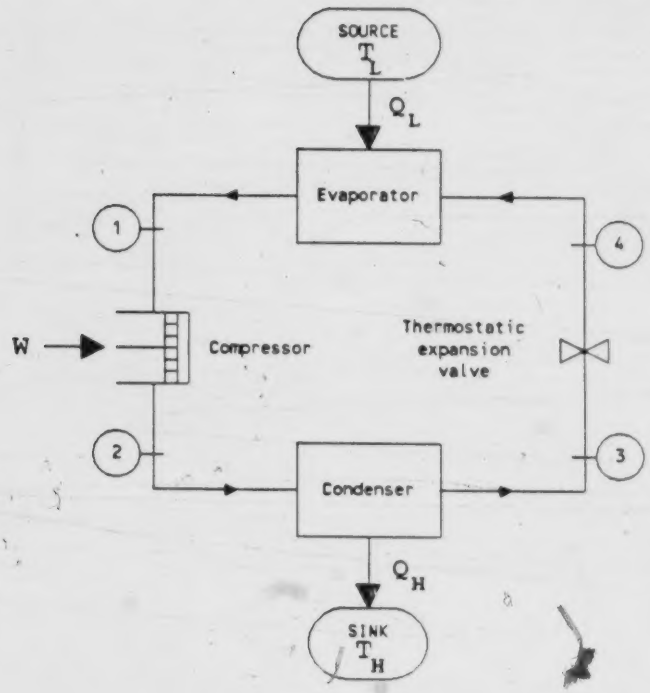


Fig.1.1a: Basic heat pump components

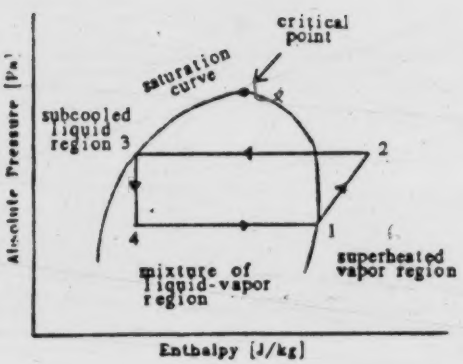


Fig.1.1b: Ideal vapor-compression cycle.

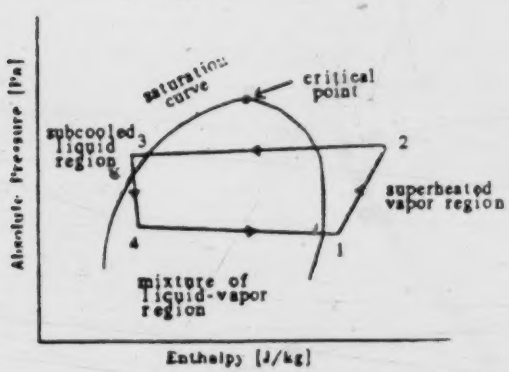


Fig.1.1c: Heat pump working cycle.

CHAPTER 2
INSTRUMENTATION

The IBM PC/XT plays a vital role in the computer-based heat pump system. The computer performs various tasks such as data acquisition, PID controller, information display and storage for off-line analysis. An interface is needed between the computer and the heat pump in order to make them properly functional.

2.1 The heat pump

The heat pump shown in Fig.2.1 basically consists of a thermostatic expansion valve, a dry, coil-type evaporator, a hermetic compressor and a water storage tank as a condenser. The principle of operation is based upon the vapor-compression cycle shown in Fig.1.1c. The cycle is characterized by four processes: expansion, evaporation, compression and condensation.

a. The expansion process: Liquid refrigerant (R 22) enters the thermostatic expansion valve at very high pressure. The valve itself is an active flow control device which uses feedback from the remote sensing bulb to regulate the mass flow rate of the refrigerant into the evaporator, thus controlling the superheat at the suction inlet to the compressor. The valve also creates a pressure drop from the high pressure side to the low pressure side of the system. The drop in pressure causes some of the refrigerant to immediately flash to gas. The latent heat required for evaporation is taken from the flowing mixture, thus cooling it. The refrigerant leaves the valve as a

liquid-vapor mixture in the saturated state, at low temperature and pressure.

The valve is factory set for a specific static superheat; however, other superheat settings can be obtained by manually adjusting the stem on the valve. It is a common practice to have the refrigerant leave the evaporator as superheated vapor in order to protect the compressor against liquid refrigerant.

b. The evaporation process: The evaporator or heat exchanger is used to provide a continual and efficient transfer of heat from a low temperature source, water in this case, to the refrigerant fluid. The source is a mixture of hot and cold water coming from the tap. While the source flow rate is maintained essentially constant by a flow regulator, its temperature can be varied by manipulating a motorized valve located at the hot water supply outlet. Accordingly, the heat pump can be tested over a wide range of source temperatures, with the rate of heat tranferred from the source to the refrigerant given by (1.4). The evaporator is constructed as a tube-in-tube heat exchanger where both fluids enter and leave at opposite ends. This arrangement is called counterflow as opposed to parallel flow where both fluids flow through the evaporator in the same direction.

The refrigerant enters the evaporator as a liquid-vapor mixture at low temperature and pressure. This is the result of its expansion through the thermostatic expansion valve. Since the water is at a higher temperature than that of the refrigerant, heat will be transferred from it through the tube wall to the refrigerant. As the refrigerant travels through the evaporator, it picks up heat from the

water, boils and gradually evaporates until it exits the evaporator as superheated vapor. As previously indicated, the thermostatic expansion valve regulates the refrigerant flow through the evaporator to ensure that the evaporator is neither starved nor overfed and that a sufficient amount of superheat is maintained in order to protect the compressor from being damaged by the entry of liquid refrigerant . To further guarantee the safe operation of the compressor, an accumulator installed in the suction line between the evaporator and the compressor is used as a liquid refrigerant trap.

c. The compression process: The compressor is a 3-phase, constant speed, hermetic type. Its primary function is to increase the pressure of vapor refrigerant in the suction line to a pressure at which the vapor can be condensed. Being of hermetic type, both the compressor and motor are integral on one shaft and housed in a pressure-sealed casing, thus posing no problem of leakage. The motor is cooled by vapor refrigerant on the suction side and it provides work needed for compression.

The refrigerant in the superheated vapor state leaves the compressor at very high temperature and pressure. The high discharge pressure provides the energy required to circulate the refrigerant against frictional resistance through the piping and equipment. The compressor is also equiped with a low/high pressure safety cutout control circuitry which stops the compressor when either the suction pressure (P_L) is too low or the discharge pressure (P_H) is too high. In addition, a crankcase heater is used to prevent refrigerant migration to the crankcase at shutdown, thus avoiding dilution of the

lubricating oil.

d. The condensation process: The useful heat energy output (Q_H) is released in the condenser where the superheated vapor refrigerant condenses at high discharge pressure produced by the compressor. The condenser is a well-stirred water storage tank that contains two submerged copper coils connected in series. As the refrigerant flows through the coils, it condenses, transferring heat to the cooling water, thus raising the temperature inside the tank. The rate of heat transfer to the condenser is given by (1.1c).

The tank also has a water inlet and outlet. The inlet flow rate of cold water is controlled by a motorized valve. Since the tank has a constant capacity, the inlet and outlet flow rates will be the same at steady state. The heat load on the heat pump can therefore be simulated by changing the cold water flow rate into the tank. The rate of heat removed from the condenser is given by (1.5).

The refrigerant now leaves the condenser as either saturated or subcooled liquid at low temperature and high pressure, and it is ready for the expansion process. The heat pump cycle is thus completed.

2.2 Measuring sensors and transducers

In order to implement a computer-based system to control the heat pump, physical quantities such as temperature, pressure, and flow rate are measured and then converted to a form, electrical voltage signal in this case, suitable for transmitting to the data acquisition system (DAS).

2.2.1 Temperature measurement

All metals produce a positive change in resistance for a positive change in temperature. This serves as the main function for the resistance temperature detector (RTD). The most common RTD's are made of either platinum, nickel or nickel alloys since these metals possess very high resistivity per circular mil foot; however, platinum RTD's are the preferred choice because they can withstand high temperatures while maintaining excellent stability and showing less susceptibility to contamination.

There are 8 RTD's, labelled T_1 to T_8, to measure temperatures at various points of the heat pump (Fig.2.1). Each of them is connected to a three-wire bridge whose configuration is shown in Fig.2.2a. The method so employed can minimize the effect of lead impedances, thus reducing measurement errors [15].

From Fig.2.2a one obtains:

$$V_o = V_{ref}\left[\frac{R_{TD}}{R_3 + R_{TD}}\right] - V_{ref}\left[\frac{R_2}{R_1 + R_2}\right] \quad (2.1)$$

For $R_1 = R_2 = 100\ \Omega$ (2.1) becomes :

$$V_o = V_{ref}\left[\frac{R_{TD}}{R_3 + R_{TD}}\right] - \frac{1}{2}V_{ref} \quad (2.2)$$

One can balance the bridge i.e $V_o = 0$ Volt at 0°C by manually adjusting R_3 until it equals R_{TD}. When the value of R_3 is fixed, V_o become a function of R_{TD} only. (2.2) is a non-linear relationship between the change in R_{TD}, or the temperature change, and the bridge output voltage.

For $V_{ref} = 1V$, V_o is a small signal and it is amplified before

being sent to the DAS. Fig.2.2b and 2.2c show the overall connection between the temperature sensors and the DAS.

2.2.2 Pressure measurement

Two gauge pressure transducers of piezoelectric type located at either end of the compressor are used to measure the discharge (P_H) and suction pressure (P_L), or the high and low side of the heat pump, respectively. One provides a measurement range of 0-100 psi and the other 0-500 psi. Both devices require DC power supply (12V) and give output voltage in the 1-6V range. The output signals are filtered before being transmitted to the DAS. No amplification is needed (Fig.2.3a and 2.3b).

2.2.3 Flow measurement

a. Source flow rate: The flow regulator (Fig.2.1) maintains the water flow rate into the evaporator essentially constant. This flow rate is measured with a differential pressure transducer. The device is a solid state piezoresistive type in the range of 0-15 psi, and it requires DC power supply (12V). One port (P1) is vented to atmospheric pressure and the other (P2) to the water inlet line (Fig.2.4). The DC voltage output (1-6V) of the transducer is filtered before being transmitted to the DAS.

b. Inlet flow rate to the condenser: The inlet flow rate of cold water to the storage tank is measured with a paddle wheel flow sensor. Since the frequency of the output signal of the sensor is linearly proportional to the velocity of the fluid, a frequency-demodulating

circuitry is needed before passing the signal to the DAS (Fig.2.5).

In Fig.2.5, the comparator converts the sinusoidal output of the flow sensor to a TTL-compatible square wave which then becomes the input of the frequency to voltage converter. The converter is designed to give a full scale 1 Volt output for an input frequency range from 0 to 100 Hz. The output signal is filtered before going to the DAS.

2.2.4 Power measurement

With the assumption of a balanced three-phase motor, the power consumed by the compressor is measured by looping one of its phase lines through the sensor. The device senses the line current and produces the corresponding DC output voltage (Fig.2.6). The number of loopings, n, increases sensitivity of the transducer by a factor n, but at the same time it reduces the maximum allowable current to be sensed by the same factor.

2.2.5 On/Off switching of the compressor

The compressor can be turned on/off from the keyboard. A switching relay connects a digital I/O line from the computer to the electrical control panel of the compressor. No driving stage for the relay is needed (Fig.2.7).

2.2.6 Final control element: The motorized valve

As previously mentioned, it is necessary to study the performance of the heat pump at various operating conditions. A change in operating conditions might occur as a result of a change in either the

heat load or the source temperature. The heat load can be changed by varying the cold water inlet flow rate into the storage tank, and the source temperature by changing the hot water flow rate into the mixture.

The permanent magnet DC motor is selected as the drive for both valves. Its advantages include low cost, compact size due to the replacement of the wound field with permanent magnets, and linear speed/torque characteristics. The rotation can be reversed simply by interchanging the two leads.

The governing equations for the permanent magnet motor are:

$$RPM = k\frac{V_a - I_a R_a}{\Phi} \tag{2.3}$$

$$Torque = k\Phi I_a \tag{2.4}$$

It is desirable to stop the motor when the valve is either completely opened or completely closed, thus preventing any possible damage done to the equipment. Since the motor has a fixed field flux (Φ), the speed of the motor can be controlled by inserting a variable power resistor in series with the motor. This method is often referred to as armature resistance control. The external resistor introduces a a voltage drop which is proportional to the current in the circuit. As the motor load increases, both the current I_a and the voltage across the external resistor increase, whereas the armature voltage V_a decreases. It follows in (2.3) that the speed of the motor will decrease with an increase in the load. The value of the external resistor has been experimentally selected such that the speed reduces to zero when the motor is about to be stalled. This condition occurs

when the valve is either fully open or fully closed.

Although the design is simple, an inherent drawback is the inability to disconnect the power supply once the motor stops. To get around this problem, a current-sensing resistor is inserted in the circuit. As the motor is running, its current is constantly monitored by the computer. When a stalled condition occurs, manifested by an sharp increase in the motor current, the computer will issue a control signal to disconnect the motor from its power supply.

Relays are used as switching devices under computer control. They can be activated by control logic from the computer through digital I/O ports available on the Lab Master DMA. For the DPDT relay type, a driving stage required. Fig.2.8 and 2.9 show the complete circuitry for both motorized valves.

2.3 Calibration

With the exception of the power sensor, all other sensors are calibrated. The data obtained are curve fitted into polynomials (Table 1) to be stored in the interrupt service routine *Handler()*.

For the power sensor, the line current to be sensed is related to the three-phase power by referring to a look-up table established from the results of an experiment where a true three-phase power meter is used and the compressor load is varied. This experiment remains to be done.

2.4 The data acquisition system (DAS)

The Scientific Solutions Lab Master DMA [16] consists of a mother

board and a daughter board interconnected by a 50-pin ribbon cable. The mother board is installed into one of the expansion slots inside the IBM PC/XT and handles the digital to analog conversion, timing/ counting and parallel I/O functions. On the other hand, the daughter board, housed in its own cabinet, handles the analog to digital conversion. The Lab Master DMA also provides features like software programmable gain (PGL) and direct memory access (DMA) where the A/D output can be transferred directly into the computer's memory without passing through the CPU.

a. The daughter board: The analog to digital (A/D) converter is located on the daughter board. Several options in terms of input type, input range, level of gain, mode of conversion, capability to sweep through a set of channels without software having to select the input channel for each conversion (auto-incrementing mode) as well as digital output format, are available through jumper and switch settings.

The daughter board has been configured to provide 16 single-ended, bipolar analog input channels with input range from -10V to +10V. The input signals are single-ended i.e all referenced to the same common ground. The normal mode of A/D conversion is selected where the three processes of channel selection, sampling and holding, and A/D conversion are performed in sequential order. Also, channel 15 is set by switches to be the last channel to be sampled in the auto-incrementing sequence when this option is enabled by software.

b. The mother board: The mother board contains two 12-bit digital

to analog (D/A) converters, a timing controller that provides five 16-bit counters, and 24 parallel digital I/O ports that can be interfaced to external devices . The board also has built-in circuitry to generate hardware interrupts to the computer's CPU.

The Lab Master DMA has been configured to be in I/O mapped mode with the DMA feature disabled. The base I/O address is set by the switch settings at 0710 hexadecimal or 1808 decimal. From this base address, 16 consecutive I/O memory locations or ports can be accessed by software. At sampling times, counter 1 is used to generate hardware interrupt requests to the CPU via IRQ2 on the 8259A Interrupt Controller. Due to their on-off nature, switching relays under computer control are interfaced to the digital I/O lines configured as output ports in place of the D/A converters. In addition, output line drivers in the form of TTL inverters are inserted between the I/O ports and the relays.

2.5 The computer IBM PC/XT

The heart of the IBM PC/XT is the Intel 8088 microprocessor. Being a 16-bit machine, the 8088 uses only an 8-bit data bus and it operates at 4.77 Mhz provided by the 8284A clock generator. Among other support chips, it is worth to mention the role of the 8259A Interrupt Controller chip. The 8259A supports 8 prioritized hardware interrupt levels (IRQ0 to IRQ7), of which IRQ2 is assigned to the Lab Master DMA. The other IRQ's have already been reserved for peripheral devices such as the keyboard, printer etc. To initiate the A/D conversion, counter 1 on the mother board generates a hardware

interrupt request (IRQ2) to the computer. Since this request has the highest priority as set by software, it is transferred by the 8259A to the CPU for attention.

The IBM PC/XT is also equiped with a 8087 Math coprocessor, a 10 Megabyte hard disk, and a Color/ Graphics Display Adapter. Fig.2.10 shows the block diagram of the computer-controlled heat pump system.

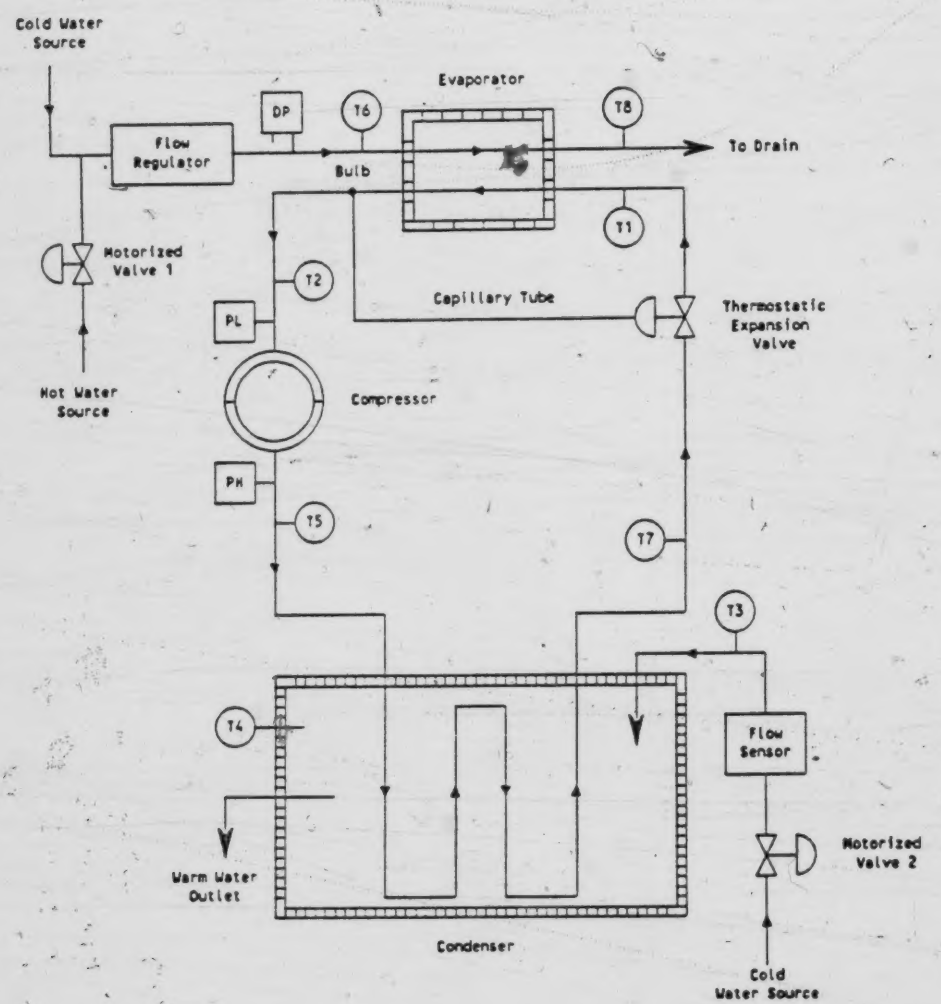


Fig.2.1: Heat Pump Block Diagram

Fig.2.2b Temperature measurement

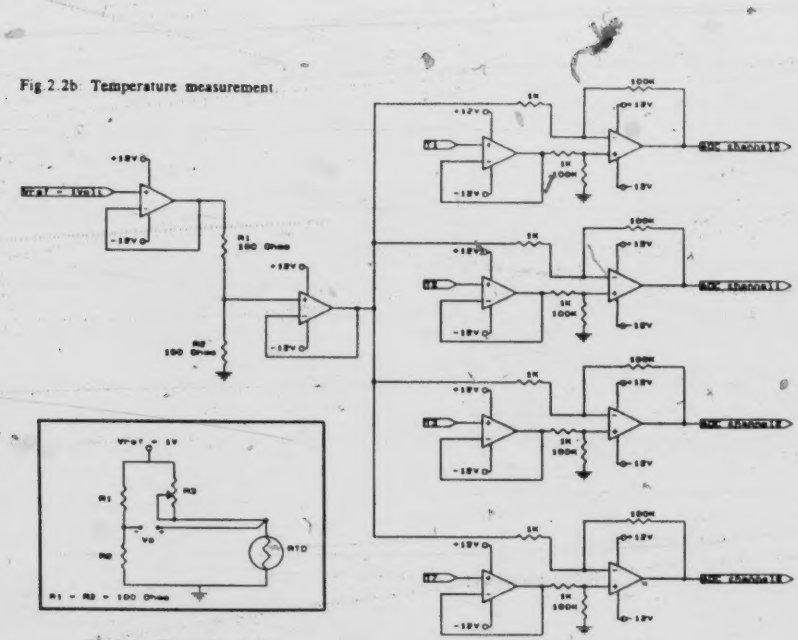


Fig.2.2a Three-wire bridge

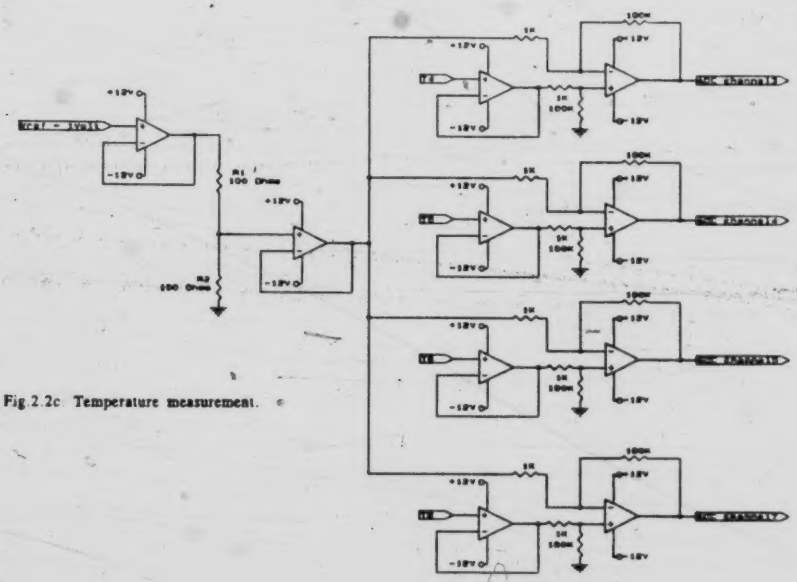


Fig.2.2c Temperature measurement.

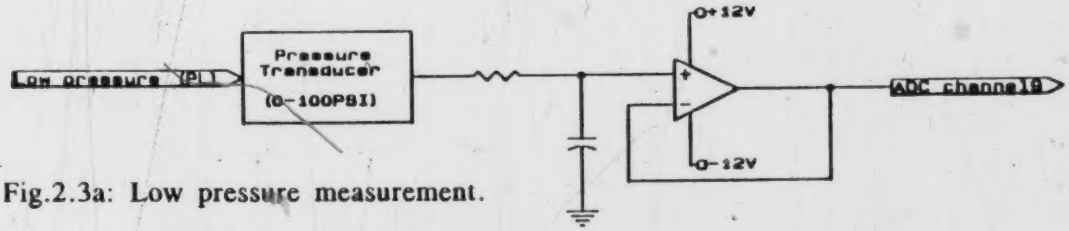


Fig.2.3a: Low pressure measurement.

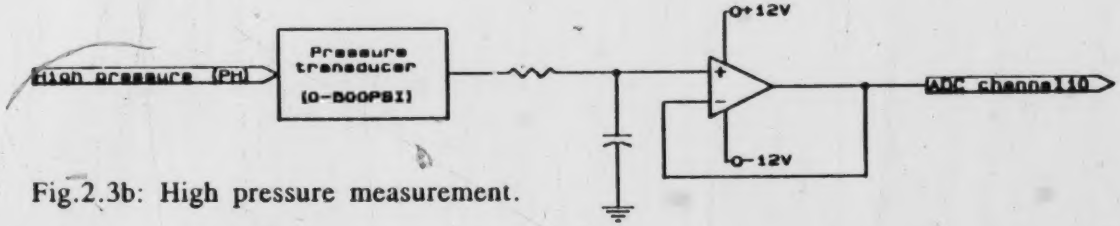


Fig.2.3b: High pressure measurement.

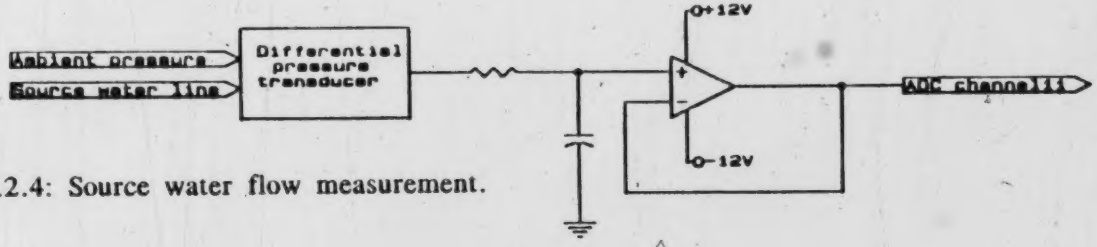


Fig.2.4: Source water flow measurement.

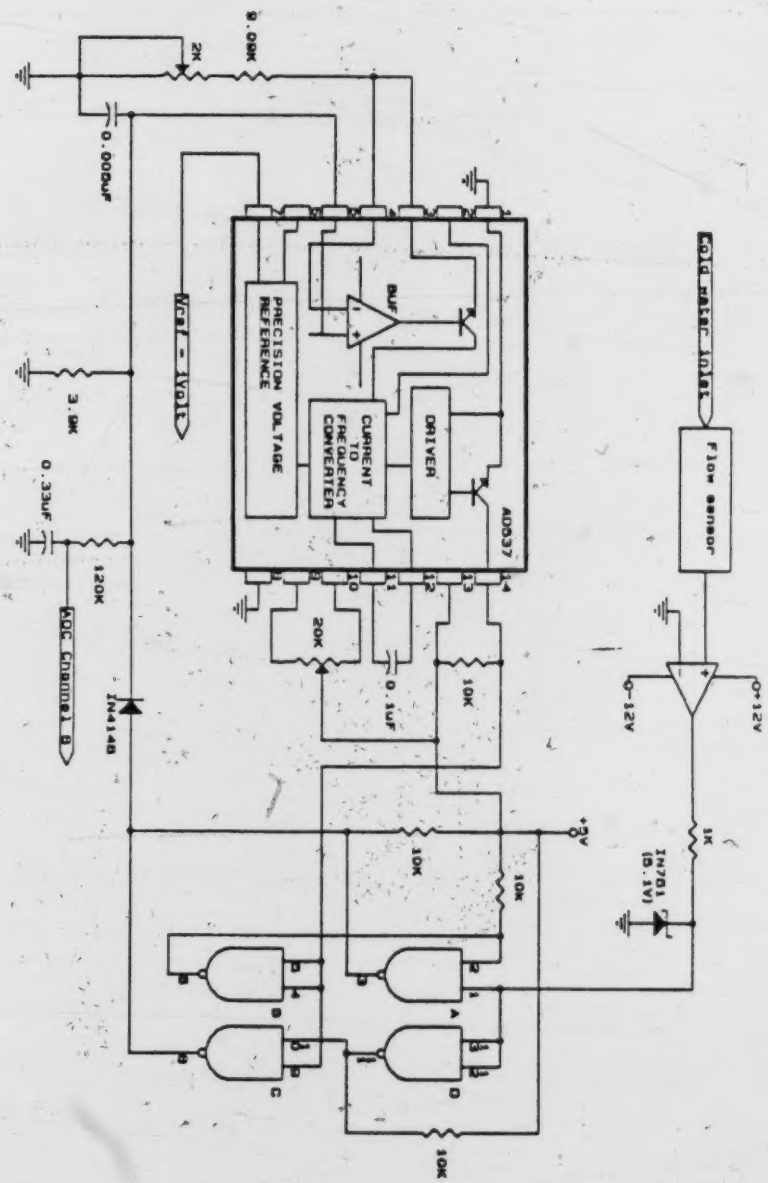


Fig.2.5: Cold water flow measurement.

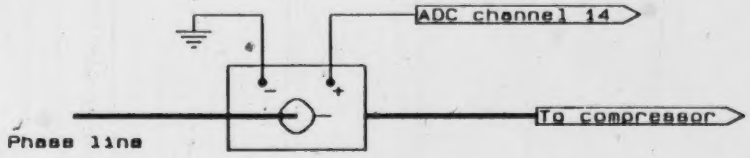


Fig.2.6: Power sensor.

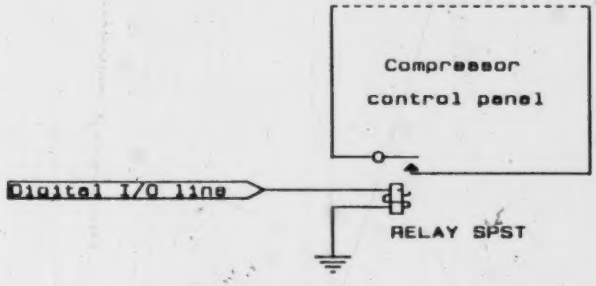


Fig.2.7: Compressor switching circuitry.

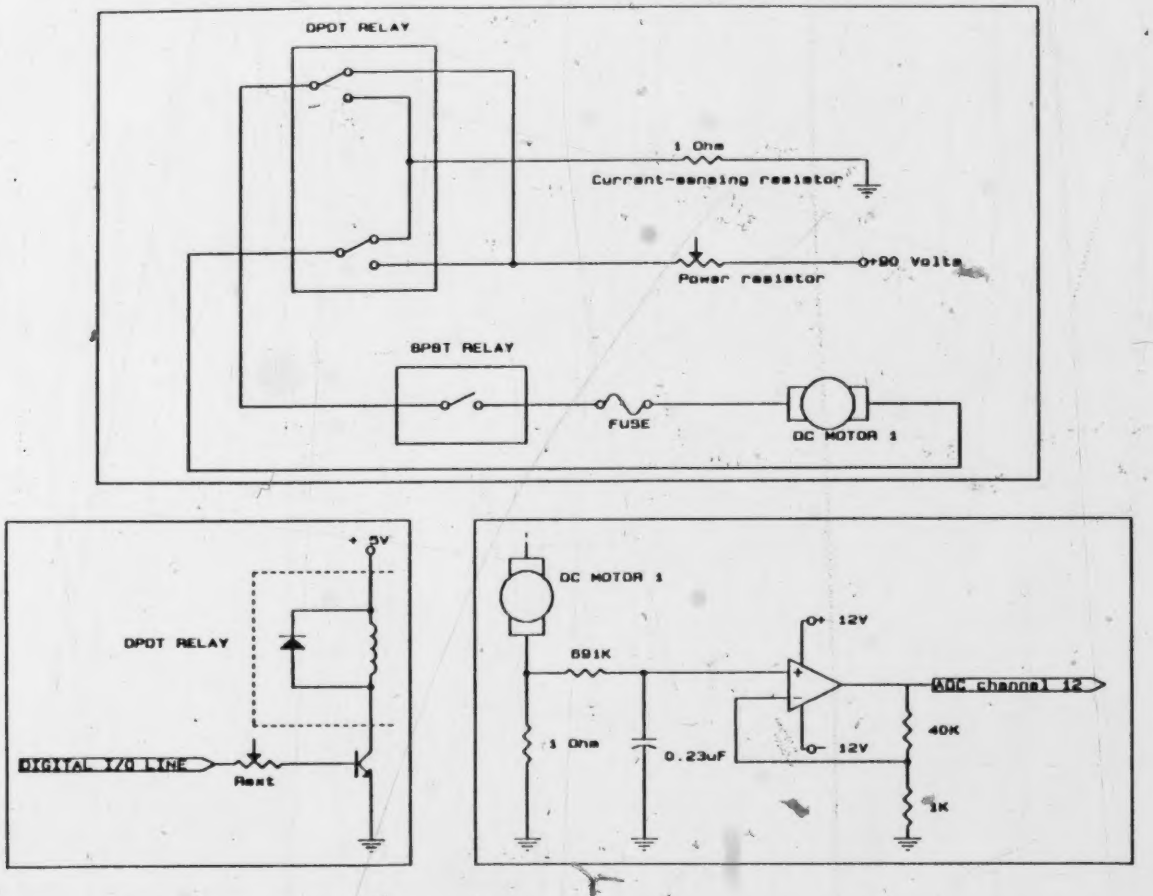


Fig.2.8: Circuit diagram of motorized valve 1.

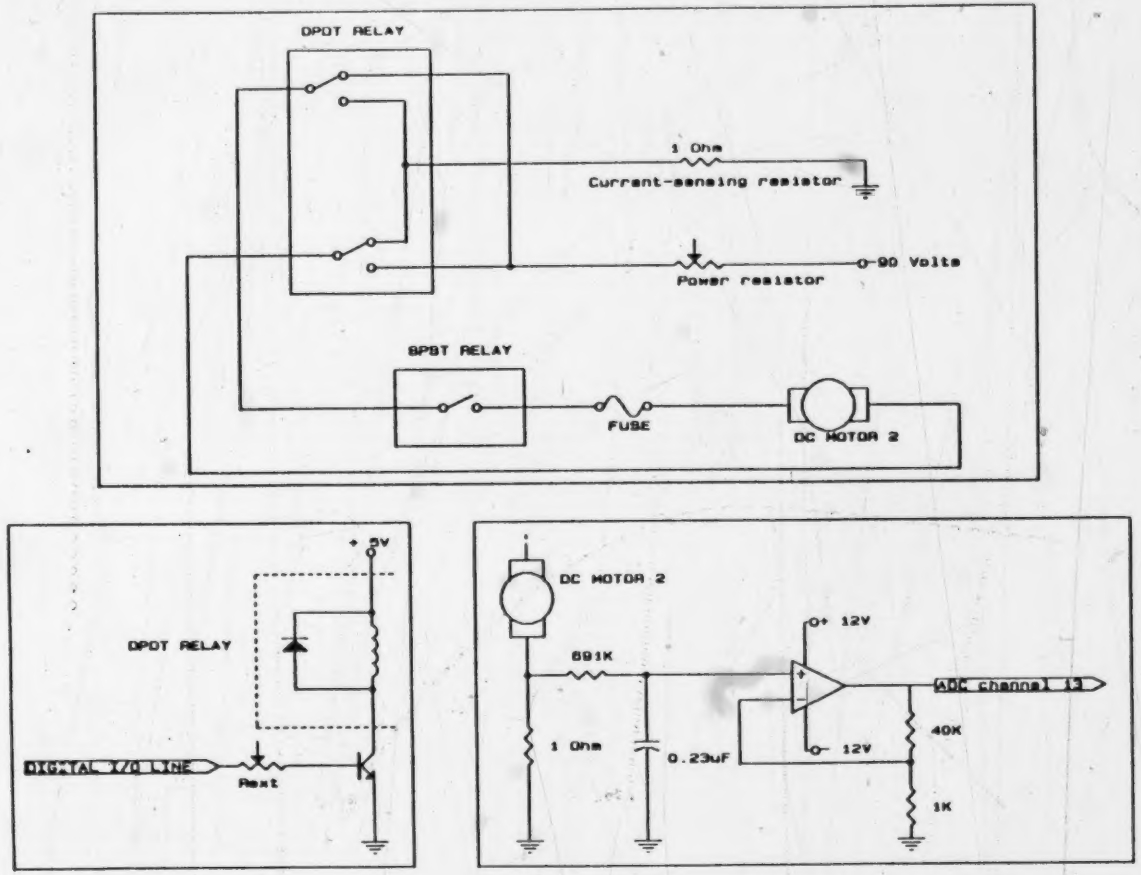


Fig.2.9: Circuit diagram of motorized valve 2.

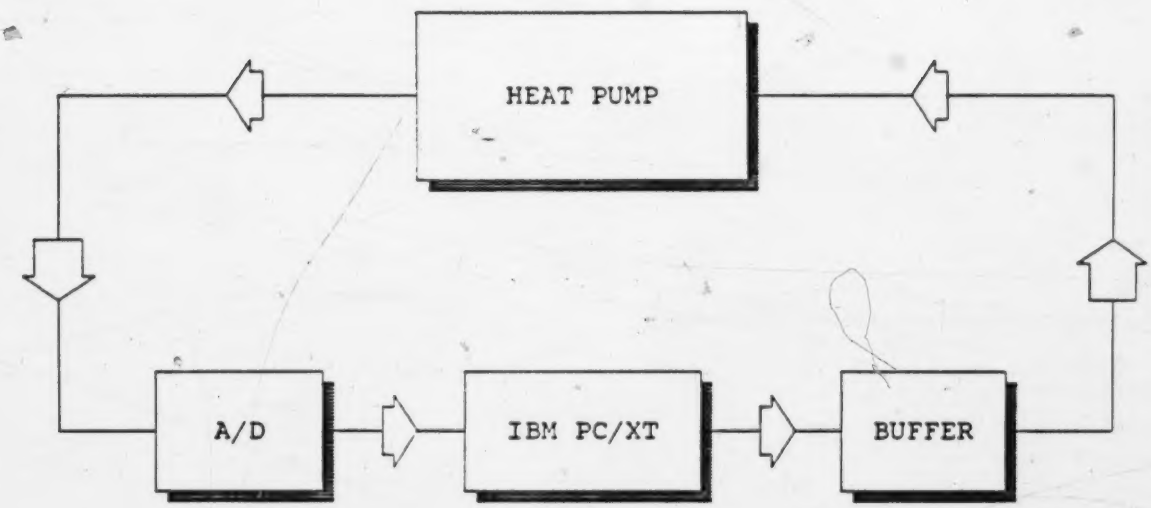


Fig.2.10: COMPUTER-CONTROLLED HEAT PUMP SYSTEM

A/D CHANNEL	SENSOR	UNIT	CURVE-FITTED POLYNOMIAL
0	T1	°C	Temp1 = ((0.0000105*tp + 0.0014222)*tp + 1.065817)*tp + 0.0091276 where tp = channel[0]*10 (mV)
1	T2	°C	Temp2 = ((0.000009*tp + 0.0015722)*tp + 1.066374)*tp - 0.1092891 where tp = channel[1]*10 (mV)
2	T3	°C	Temp3 = ((0.0000053*tp + 0.0020518)*tp + 1.055428)*tp - 0.2182054 where tp = channel[2]*10 (mV)
3	T4	°C	Temp4 = ((0.0105*tp + 0.1345451)*tp + 10.89598)*tp + 0.00983 where tp = channel[3] (V)
4	T5	°C	Temp5 = ((0.0000071*tp + 0.0019428)*tp + 1.051965)*tp + 0.418166 where tp = channel[4]*10 (mV)
5	T6	°C	Temp6 = ((0.0000067*tp + 0.0018578)*tp + 1.046927)*tp - 0.060407 where tp = channel[5]*10 (mV)
6	T7	°C	Temp7 = ((0.0375811*tp - 0.1055912)*tp + 11.52742)*tp - 0.1331686 where tp = channel[6] (V)
7	T8	°C	Temp8 = ((0.0113451*tp + 0.1541135)*tp + 11.45268)*tp + 0.8992917 where tp = channel[7] (V)
8	WFLOW	L/Min	Wflow = 34.496214*tp + 0.948245 where tp = channel[8] (V)
9	PL	psi	Lpress = 20.170923*tp - 22.099904 where tp = channel[9] (V)
10	PH	psi	Hpress = 97.26633*tp - 87.959346 where tp = channel[10] (V)
11	DP	L/Min	Eva_flow = ((3.9846*tp - 35.741)*tp + 116.721)*tp - 110.417 where tp = channel[11] (V)
12	SEN1	mA	N/A (Current sensing valve 1)
13	SEN2	mA	N/A (current sensing valve 2)
14	POWER	J/s	Remains to be done
15			Unused

Table 1: Calibration.

CHAPTER 3
SOFTWARE DEVELOPMENT

With the exception of a small portion of code written in assembly language, the program is mainly written in Microsoft C. C is a general-purpose, structured programming language. However, unlike Pascal or Ada, it is often called a middle level language which incorporates the functionalism of the assembly language, bit manipulation for instance. C code is very compact, easy to read and portable to other environments. Though powerful and efficient, C is a relatively small language. It does not have built-in functions for tasks like input and output, memory allocation, string manipulation etc. Instead, C relies entirely on run-time libraries to perform such tasks. The run-time library provides routines for accessing the operating system like MS-DOS and for common programming tasks frequently encountered.

3.1 Program structure

Originally written by Tung [17], the program has been revised to suit the current application, where such features as multiple sampling rates and extensive operator interface are required. Basically, it follows a modular structure for easy modification and enhancement. All major tasks are broken down into smaller, specific ones to be carried out by routines. In addition, variables are generally defined to be external so that communication between them and routines can be done easily without the need for argument lists.

The program consists of two modules which are separately compiled and then linked together into one executable file.

The program code mainly resides in the C module, whose structure is shown in Fig.3.1, whereas the assembly module contains only a small routine in assembly language which serves as the interrupt handler for hardware interrupts sent by the DAS to the CPU.

The program consists of one main menu, PID screen, Display screen, and Help screen. On the main menu, the operator can select a particular screen or he can terminate the program and return to MS-DOS. Screen switching can be accomplished with the left/right arrow key and only from Display screen can one go back to the main menu. Communication between the operator and the computer is made possible through the keyboard, and all the three screens share a common interface, the *I_O()* routine, which calls appropriate routines to carry out the task.

Fig.3.2 shows the block diagram of the menu-driven program and Table 2 the keyboard assignment. Associated with the keyboard assignment are control and status flags. Control flags are used for controlling the program flow and status flags for indicating the status of external peripherals such as the DAS. As a particular key is pressed, certain control and status flags are set or reset. *I_OFLAG* and *STOP* are both examples of control flags, either of which , when set to 1, will cause the program flow to exit *I_O()* routine.

Before the program can be examined in detail, it is necessary to consider the IBM PC/XT memory and its interrupt structure.

3.2 System memory

Since it has a 20-bit address line, the 8088 can address 2^{20} or 1,024K bytes of memory (address 00000 to FFFFFhex). The first 640K bytes (address 00000 to 9FFFFhex) are provided by RAM on the system board and used as working memory. The Color/Graphics Adapter (CGA) installed in one of the expansion slots provides a 16K-byte video memory. Although physically located on the adapter card, the video memory is logically a part of the computer's 1,024K-byte addressable memory space with its its address starting at B8000hex [18].

System memory addresses can be represented by segment-offset pairs. For the video memory, the starting location has the segment value of B800hex and the offset value of 0000. Given any segment-offset pair, the corresponding address can be calculated from the formula [19]:

Absolute address = (0x10 • segment value) + offset

3.2.1 Accessing system memory for graphic display

Access to video memory can be done directly or by services provided by BIOS interrupt 10hex.

a. Direct memory access: In high-resolution text mode, a screen can contain 80x25 (2000) characters, each of which occupies 2 bytes in memory. The first byte, stored at an even address, holds the character code, and the second byte its attribute, which specifies color code and display mode. The 16K-byte video memory can hold up to 4 pages or screens of text. Fig.3.3 shows the mapping between video data and and their locations on the screen.

In C language, direct memory access can be accomplished with far pointers. A far pointer is a 32-bit pointer, whose two high-order bytes contain the segment address and two low-order bytes the offset address.

The following code shows the implementation of far pointers in C:

```
unsigned int far *ptr;                    /* ptr is a 32-bit pointer */
ptr = (int far *) (0xB800L << 16);/* ptr points to B8000 */
```

To understand how a message can be displayed at a specific location on the screen, one can examine routine *screenf(mess, row, column, color)*, where *mess* is the message to be displayed in *color*, at location specified by *row* and *column*.

```
#Define disbuff 0xB8000000                /* 32-bit constant */
                :
                :
void screenf(mess,row,column,color)
char mess[], color;
int row, column;
{
char far *position;                       /* far pointer */
int count = 0;
/* count counts number of characters in mess */
position = (char far *)((80*(row - 1)+ column -1)*2 + disbuff);
/* position points to an even memory location that maps to point
                                          (row,column) on the screen */
while (mess[count] != '\0')
  {
     (*position++) = mess[count++];
     /* put character at address pointed to by position then
        increment position to point to next location in memory,
        and count to next character in message */
     (*position++) = color;
     /* put attribute color at address pointed to by position and
        increment position to point to next location on screen */
  }
}
```

One can directly access video memory to draw boxes, vertical and horizontal lines, color the whole screen or a portion of it, and

display a message with or without specified color, using *box()*, *vertical_line()*, *horizontal_line()*, *colorf()*, *screenf()*, and *charf()*routines, respectively.

b. BIOS interrupt 10hex: This software interrupt provides a wide range of video functions such as setting display mode (function 0), scrolling the active page up (function 06hex). Interrupt 10hex can be invoked using *int86()* library function, with the video function number being placed in register AH.

The main advantages of using direct memory access instead of BIOS interrupt 10hex are increased speed and increased control of the screen; however, programming becomes more involved due to inherent details and the program becomes machine-dependent once specific addresses are incorporated into it.

c. Cursor control: It is necessary to display measured variables at specified locations on the screen. There are various methods to control cursor positioning, the simplest of which is using the escape sequences. The *ansi.sys* file, which is provided with DOS versions 2.0 and later, contains code that transforms escape sequences to the proper routine calls. The statement *'device=ansi.sys '* must be included in the *config.sys* file for this method to work. Table 3 shows the cursor control sequences [20].

As an illustration, the following statement moves cursor to location (10,20) on the screen

```
cputs("\033[10;20H");      /* ESC in ASCII code is octal 033 */
```

3.3. Interrupts

The 8088 can handle up to 256 (0 through FFhex) different interrupt types. Interrupts may be initiated by devices external to the CPU (hardware interrupts), by software interrupt instructions (INT interrupt number), and under certain conditions (divide by zero etc.), by the CPU itself. Fig.3.5 shows the interrupt sources and Table 4 their priorities [21].

The lowest portion of system memory (address 00000 to 003FFhex) is reserved for the interrupt vector table, which may include up to 256 entries or vectors, one for each interrupt type. Each 4-byte vector in the table contains the address of the interrupt handler for that type. The base address or code segment is stored in the second 2 bytes (higher memory) while the offset from the beginning of the code segment is stored in the first 2 bytes. Since each vector is 4 bytes long, one can find the address of a particular interrupt handler by multiplying its interrupt type by 4 to obtain its corresponding vector in the table and then use DEBUG, a MS-DOS utility, to examine its contents.

3.3.1 The 8259A Programmable Interrupt Controller (PIC)

Table 5 lists the 8088 hardware interrupt sources and their normal priority setting [18]. They can be selectively masked (disabled) and their priorities adjusted by software. Hardware interrupts are handled by the Programmable Interrupt Controller (PIC). Inside the PIC are the Interrupt Request Register (IRR) to receive interrupt requests arriving on the IR0-IR7 pins, the Interrupt Mask

Register (IMR) to store all interrupt levels which are masked, the Priority Resolver to resolve any conflict resulting from the simultaneous arrival of multiple interrupt requests, and the In-Service Register (ISR) to store all interrupt levels currently being serviced.

As interrupt requests arrive, they are evaluated by the PIC, which then issues an INT signal to the CPU, if appropriate. Whether or not the CPU honors the request depends upon the current state of the interrupt enable flag (IF). If IF is cleared, the CPU will ignore the request and carries on its normal operation. IF can be set or cleared with STI and CLI instructions, respectively. To acknowledge the request, the CPU will first complete executing the current instruction and it then issues two INTA pulses. Upon the arrival of the first pulse, the PIC will temporarily disable the IRR latches so that further requests on the IR0-IR7 lines will be ignored. This condition will be maintained until the end of the second pulse. The first pulse also causes the appropriate ISR to be set and the corresponding IRR bit to be cleared. During the second pulse, the PIC returns an 8-bit data which indicates the interrupt type associated with the device requesting service. The CPU develops this byte into a 10-bit address at which the interrupt vector containing the address of the interrupt handler is located. Also, at the end of the second pulse, the ISR bit, which is set by the first pulse, will be cleared if the automatic end of interrupt (AEOI) bit in ICW4 is 1; otherwise, the end of interrupt command (EOI) has to be issued to clear it.

Before calling the interrupt handler, the CPU saves the flags and

the address of the next instruction, to which the CPU will return after servicing the interrupt, on the stack. The CPU then clears IF to disable further requests from the PIC. Control is now transferred to the handler. First of all, the handler usually resets IF with the STI instruction to allow interrupt requests of higher priority to occur. It saves any register to be used, and then services the device. Upon the completion of the task, it issues the EOI command, retrieves all the saved register contents from the stack, and finally, it issues the IRET instruction which restores the original flags, the code segment (CS), and the instruction pointer (IP). The CPU now returns to the point where it has left off [22].

a. Programming the PIC: The PIC can be addressed through I/O port addresses 20hex and 21hex, and it accepts two types of command words: Initialization Command Words (ICWs) and Operation Command Words (OCWs). Table 6 summarizes the addressing structure of the PIC.

The PIC has to be initialized before it can be programmed to operate in various interrupt modes using OCWs. The proper initialization sequence is shown in Flow chart 1.

In the IBM PC/XT, the initialization of the PIC is transparent to users. It is done by the BIOS at power-on time, with ICW1 = 13hex (edge-triggered, single PIC, ICW4 included), ICW2 = 08hex (interrupt type 8 - 0Fhex), and ICW4 = 0Dhex (buffered, 8088 mode, normal end of interrupt). ICW3 is not used since the PIC is the only interrupt controller in the system as indicated in ICW1 [18]

Once the initialization is completed, the PIC is ready to accept interrupt requests at its input lines. During program execution, its

mode of operation can be dynamically changed with OCWs. In the current application, hardware interrupts are not allowed while a critical section of code such as the updating of the controller's parameters, is executed.

```
outp(0x21, 0xff);        /* OCW1 = 11111111: all hardware interrupts
                            are disabled i.e IMS bits are set to 1 */
        .
        .
        .
(critical section of code is executed)
        .
        .
        .
outp(0x21,0);            /* IMS bits are cleared */
```

When the DAS is active, its interrupt requests via IRQ2 are assigned highest priority, whereas interrupts from other sources except the keyboard are temporarily disabled in order to minimize real time overhead in handling multi-level priority interrupts. The following code in routine *starter()* specifically masks all interrupt requests except those from the DAS and the keyboard, and it gives top priority to IRQ2.

```
outp(0x21, 0xf9);        /* ocw1 = 11111001: mask all interrupts
                            except IRQ1, IRQ2 */
outp(ox20, oxe1);        /* OCW2 = 11100001: specific rotation of
                            priority, IRQ1 has lowest priority */
```

To allow all hardware interrupts and the resumption of normal priority setting, as is the case when the program terminates and returns to DOS, the following code can be executed.

```
outp(0x21, 0);           /* OCW1 = 0; clears the IMR register */
outp(0x20, 0xe7);        /* OCW2 = 11100111: IRQ7 becomes lowest
                            priority */
```

3.3.2 Setting up the interrupt vector for IRQ2 interrupt handler

One can use either direct memory access approach or facilities provided by MS-DOS to install the DAS interrupt handler.

a. Direct memory access: The vector that contains the address of the DAS interrupt handler starts at absolute location 0A*4 (28hex) in the interrupt vector table. A far pointer is used to point to this location.

```
#Define FP_INT_ADDR(seg,off)  (insigned int far *)  (((unsigned\
                                long) seg) << 16) | (unsigned long) off)

                    :

unsigned int far *vector;          /* 32-bit pointer to type int */
vector = FP_INT_ADDR(0,0x0A * 4);/* Vector points to absolute
                                address 00028H in the vector table */
*(vector + 1) = cs();              /* program code segment returned by
                                   cs() is stored at the high 2 bytes
                                   pointed to by vector */
*(vector) = (unsigned int) START;  /* the offset of the
                                   interrupt handler routine
                                   relative to the program code
                                   segment is stored at the lower
                                   2 bytes pointed to by vector */
```

b. MS-DOS interrupt 21hex: Functions 25hex (set interrupt vector), 35hex (get interrupt vector) and 31hex (terminate and stay resident) available through interrupt 21hex can be used to install well-behaved interrupt handlers. In the case of the DAS handler, only function 25hex is needed since the program does not stay resident upon termination and interrupt requests from the DAS are via IRQ2, the only input line to the PIC not occupied by any particular peripheral. In principle, DAS interrupts can be tied to any of the IRQs lines; however, the choice of IRQ2 is convenient from the programming point of view because any other choice requires that the current contents

of the interrupt vector be saved before being modified, and then restored when the program terminates.

```
extern START;                          /* name of far procedure */
        :
        :
void setuphandler()
{
    union REGS inregs, outregs;
    struct SREG segregs;
    inregs.h.ah  = 0x25;                    /* Function 25H */
    inreg.h.al   = 0x0A;                    /* IRQ2 interrupt level */
    inregs.x.dx  = FP_OFF((unsigned int *)(START));
                                    /* Get the offset of the handler
                                       routine */
    inregs.x.ds  = FP_SEG((unsigned int *)(START));
                                    /* get the code segment of the
                                       hander routine */
    indosx(&inregs, &outregs, &segregs);    /* invoke INT 21H */
}
```

3.3.3.Servicing DAS interrupts

In the heat pump system, only the source temperature (T_o) and the cold water flow rate into the condenser (F_o) need to be sampled at a fast rate because they are under automatic control. Other measured variables can be sampled at a much slower rate in order to reduce real time overhead. Accordingly, there are two sampling periods , T_{s1} and T_{s2}, for the DAS. Their relationship is conveniently chosen to be:

$$T_{s2} = n*T_{s1}$$ where n is an integer.

For all practical purposes, T_{s1} is selected to be 1 sec. Fig.3.6 shows the time frame for n = 4. At T_{s1}, specific A/D channels are sampled. At T_{s2}, the auto-increment option is selected and the A/D conversion sweeps through all channels sequentially.

The interrupt handler consists of a far procedure, *START*, whose

main task is to call the external C routine *handler()* to service the DAS when interrupts arrive. In addition, it performs some housekeeping functions before and after the DAS has been serviced.

The handler is written in assembly language because C neither supports the Return from Interrupt instruction (IRET) nor provide library functions to access registers other than the segment registers. A word of memory located before the procedure stores the program data segment; this value is subsequently restored to the DS register to ensure that, while *handler()* is servicing the DAS, the program data segment is properly addressed.

The C module is compiled with the small memory model option, which limits the code and data to a 64K segment each. Compared to other memory models, the small memory model is the fastest in terms of speed. The underlying reason is that only a 16-bit pointer is required to address any item in the program, as opposed to a 32-bit pointer for a full segmented address. To obtain the desired C-assembly interface between the two modules, the assembly module begins with the required segment definitions, which are the same as the one produced by the IBM C compiler for a small memory model program [19].

MODULE 2: File *int.asm*

```
; Segment definitions
_TEXT   SEGMENT BYTE PUBLIC 'CODE'
_TEXT   ENDS
_DATA   SEGMENT WORD PUBLIC 'DATA'
_DATA   ENDS
CONST   SEGMENT WORD PUBLIC 'CONST'
CONST   ENDS
_BSS    SEGMENT WORD PUBLIC 'BSS'
_BSS    ENDS
DGROUP  GROUP   CONST, _BSS, _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP,
                ES: DGROUP
```

```
;   The   interrupt   han d ler   starts   here. Its   actual   address   is   the
; beginning of procedure START.
_TEXT    SEGMENT
    _DATA_SEG    DW  (0)   ; Location to store contents of DS

_START   PROC  FAR
         PUBLIC  _START     ; START can be accessed globally.
         EXTERN  _handler   ; handler is external to module
                 .          ; save flags and registers with
                 .          ; PUSH instructions.
         MOV     AX, CS:_DATA_SEG
         MOV     DS, AX     ;   restore   the   program   data   segment
         CALL    _handler   ; calls handler() routine
         MOV     AL,20H     ; issues EOI as required by
         OUT     20H,AL     ; the 8259A
                 .          ; restore flags and registers
                 .          ; with POP instructions
         IRET               ; issues Return from interrupt
                            ; instruction.
_START   ENDP               ; end of procedure.
_TEXT    ENDS
         END
```

3.3.4 The *handler()* routine

The primary purpose of *handler()* is to start the A/D conversion, from which the sampled data are collected and converted back into appropriate units. The flow chart of *handler()* is shown in Flow chart 2a&b.

No hardware interrupts are allowed while the channels are sampled. In addition, it is important to bear in mind that MS-DOS is not reentrant, which means that DOS functions issued through interrupt 21hex can not be invoked anywhere within the interrupt handler. Any attempt to execute, for an instance, a *printf* statement inside *handler()* will immediately halt the computer.

3.4 The data acquisition system (DAS)

The DAS resides in I/O memory space, at address 710hex. From this base address, 16 consecutive 8-bit locations or ports are reserved for it. C language provides *inp()* and *outp()* library functions to write to, and read from, these locations [16].

Available on the DAS are five 16-bit counters under the supervision of the 9513A System Timing Controller chip (STC), and 24 digital I/O lines controlled by the 8255A Peripheral Interface chip (PI). The STC is driven by a 1 MHz crystal. This basic frequency is further divided with either BCD or binary scaling option to provide four extra different frequency sources for the counters. One can access the STC via ports 718hex and 719hex, and the PI via ports 71Chex, 71Dhex, 71Ehex, and 71Fhex.

3.4.1 Programming the System Timing Controller (STC)

The DAS is configured to have counter 1 generate hardware interrupts (IRQ2) to the CPU at sampling times (Fig.3.7).

The STC has to be configured before counter 1 can be put in operation. As a normal procedure, first of all the Master Reset command is sent to the control port to clear all registers and set their outputs to a low state.

```
outp(TECMAR + 9, 0xff);
```

Next, the auto-sequencing feature is disabled to allow direct access to the registers.

```
outp(TECHMAR + 9, 0xe8);
```

The Master Mode register is then selected to configure the STC so that the time of day and the comparators are disabled, the FOUT source input is SRC1 of counter 1, FOUT divider is 1, FOUT gate is enabled, auto sequencing is disabled and BCD scaling is used.

```
outp(TECMAR + 9, 0x17);      /* select register */
outp(TECMAR + 8, 0x10);      /* output low byte */
outp(TECMAR + 8, 0xd1);      /* output high byte */
```

The configuration of the STC is done by *format()* routine. Of relevance to the current application are the disabled auto sequencing feature and the BCD scaling option. From the base frequency (1MHz) generated by the crystal, BCD scaling makes available to counter 1 four more different frequency sources (100Khz, 10KHz, 1KHz, and 100Hz). Since the maximum unsigned integer which a 16-bit counter can hold is 2^{16}- 1 or 65535, the source frequency input (F) to counter 1 will depend upon the sampling period (T_{s1}) i.e the longer the period, the lower the frequency source. Table 7 shows the frequency sources and the corresponding sampling periods [17].

For a particular T_{s1}, the corresponding count value loaded into counter 1 is calculated from the following formula:

$$Count = (T_{s1} * F) - 1$$

Before loading counter 1 with 'count' value and starting the counting process, one has to specify the mode of operation by writing to the Counter Mode register. Counter 1 is configured such that it counts on the rising edge of the pulse from the source frequency F with no gating, which means that the Arm command has to be sent to the control port to start the counting. The countdown is in binary format

with the special mode disabled. Counter 1 counts once and the active low terminal count pulse will signal the interrupt.

The following code specifies the mode of operation for counter 1, and it selects F3 as the frequency source for T_{s1} = 1sec.

```
outp(TECMAR + 9, 0x01);    /* select counter 1 mode register */
outp(TECMAR + 8, 0x05);    /* output low byte */
outp(TECMAR + 8, 0xod);    /* output high byte */
```

Counter 1 is now ready for operation. It is loaded with the count value and only when the Arm command is sent to the control port does it start the countdown. Although there are two sampling periods T_{s1} and T_{s2}, one only needs to load counter 1 with the count value corresponding to T_{s1}.

The following code loads *counter*, the count value for T_{s1}, into counter 1 and then starts the countdown. In addition, the A/D conversion mode is selected such that the auto-increment feature is disabled and that, at the end of the countdown, counter 1 will generate a high-going pulse (IRQ2) for use in interrupting the CPU.

```
outp(TECMAR + 9, 0x09);    /* select counter 1 load register */
outp(TECMAR + 8, counter && 0x00ff);  /* load low byte first */
outp(TECMAR + 8, counter >> 8);       /* then the high byte */
outp(TECMAR + 9, 0x21);               /* start count down */
outp(TECMAR + 4, 0x90);               /* control byte */
```

When the countdown is completed, counter 1 signals the CPU for attention. Since the request is of highest priority, the CPU temporarily suspends its current activity, and calls the interrupt handler, which in turn calls *handler()* to service the DAS. When invoked, the first thing *handler()* does is resetting counter 1 to *counter* and enabling the countdown for the next sampling time.

3.4.2 A/D conversion

The A/D converter accepts a bipolar analog input in the range from -10V to 10V and converts it to a digital output at the sampling time. The sampled output data can be obtained by reading its low byte first at location 715hex, and then its high byte at location 716hex. Because it is in two's complement format, the output data needs to be converted back into voltage in order to make it physically meaningful. Table 8 shows the mapping between analog inputs and digital outputs, and the two's complement to voltage conversion. Each step in the digital output corresponds to $\frac{20V}{4098} = \frac{1V}{204.8}$ [16].

Location 714hex is reserved as the control port for the A/D conversion. An 8-bit data written to this port allows the conversion mode to be selected. At T_{s1}, the control byte is 10010000, which specifies that the software gain is not used, the source to interrupt the CPU comes from counter 1 of the STC, a command is sent to start the conversion, and the auto-increment is disabled. At T_{s2}, bit 7 of the control byte is reset to enable the auto-increment.

a. The auto-increment is disabled: When this option is not used, which is the case at the sampling time T_{s1}, a specific channel can be selected for sampling by writing its channel number to port 715hex. After a command is sent to port 716hex to start the conversion, a wait state is entered until the end of conversion flag (bit 7 of the A/D status byte) is set, signalling that the digital output is now available at ports 715hex and 716hex for reading. The program detects the end of conversion by polling the A/D Status register which is

accessible via port 714hex.

The following code specifies the conversion mode and selects channel 5 for sampling.

```
outp(TECMAR + 4, 0x90);              /* control byte */
outp(TECMAR + 5, 5);                 /* selects channel 5 */
outp(TECMAR + 6, 0);                 /* start conversion */
while (inp(TECMAR + 4) < 128);       /* wait */
low = inp(TECMAR + 5);               /* read low byte */
high = inp(TECMAR + 6);              /* read high byte */
data = (high << 8) + l0w;            /* combine data */
if (data > 32767) data = data - 65536;
data = data / 204.8;                 /* convert data to voltage */
```

b. The auto-increment is enabled: At T$_{\Omega}$, all channels are sampled sequentially, with channels 0 and 15 being the first and the last in the sequence, respectively; the former is selected through software and the latter through the switch setting during the configuration of the DAS.

An initialization procedure has to be followed prior to enabling the auto-increment. The following code samples all channels in the automatic mode.

```
/* Procedure to set up the auto-increment */
outp(TECMAR + 4, 0x90);              /* control byte */
outp(TECMAR + 5, 255);
outp(TECMAR + 6, 0);
while (inp(TECMAR + 4) < 128);
outp(TECMAR + 6);
/* Set up auto-increment mode */
outp(TECMAR + 4, 0x10);              /* control byte */
outp(TECMAR + 5, 0);                 /* channel 0 is the first */
for (i = 0; i < 16; i++ )            /* channel in the sequence*/
{
     outp(TECMAR + 6, 0);            /* start conversion */
     while (inp(TECMAR + 4 < 128));  /* wait */
     low = inp(TECMAR + 5);          /* read low byte */
     high = inp(TECMAR + 6);         /* read high byte */
     data = (high << 8) + low;       /* combine data */
     if (data > 32767) data = data - 65536;
     channel[i] = data/204.8;
}
```

3.4.3 Programming the Peripheral Interface chip (PI)

Control signals to the switching relays are sent by the CPU via the digital I/O lines of the PI. Fig. 3.8 shows the block diagram of the arrangement.

The 24 digital I/O lines are accessible through ports 71Chex, 71Dhex, and 71Ehex. They are configured to be output lines by sending a value of 80hex to control port 71Fhex. In this mode of operation, output data from the PI will be latched or remain on the lines until new data is sent from the CPU. Up to 24 switching relays can be interfaced to the buffer, which behaves as an output line driver, via the industry standard 50-pin connector (J11). Since the control signal is inverted by the buffer, a low bit (0) sent by the CPU will switch the relay ON; whereas a high bit (1) will switch it OFF.

The following code will configure the PI to be in the output mode and specifically switch ON the relay connected to pin 1 of J11:

```
outp(TECMAR + 15, 0x80);        /* configure output mode */
.
outp(TECMAR + 12, 0xfe);        /* control signal = 1111 1110 */
```

This relay can later on be switched off by sending a high bit to port 71Chex.

```
outp(TECMAR + 12, 0xff);
```

3.5 PID screen

Fig.3.9 and 3.10 show the block diagrams of two feedback control loops in the heat pump system. The controlled outputs are the flow

rate (F_o) into the storage tank and the source temperature (T_o) at the inlet of the evaporator. PID controllers, commonly employed in process control, are implemented in both loops.

Both feedback loops are non-linear due to the presence of a switching relay in the forward path. The relay can be switched ON or OFF depending upon whether or not the controller output lies within the dead zone specified by *'relay dead zone'*.

The PID controller has the following features:

- Integral action is rendered inactive when the steady state error, the difference between the controlled output at steady state and the set point, falls within an acceptable range specified by *'Allowed Ess'*.
- Integral windup is prevented by *'Reset windup'*, a limit at which the integral term saturates.

Due to a significant dead time inherent in the temperature loop [23], a dead time compensator or Smith predictor, specified by *'Process gain'* and *'Time constant'*, is inserted in the loop. Compensation, however, is not needed in the flow loop because its flow rate responds almost immediately to a change in the valve position. The design of both controllers will be discussed in chapter 4.

Before parameters can be entered, their controller has to be specified with either the <F3> or <F4> key. The <F3> key is used to select the flow controller and the <F4> key the temperature controller . Next, one can use the <UP/DOWN arrow> keys to select the parameter for updating. The parameter to be updated is highlighted. Table 9 shows the control and status flags associated with PID screen.

3.6 Display screen

It is on this screen that major tasks are performed. The screen offers the following features:

a. After loading the sampling rates, the operator can activate the DAS. Data from the A/D conversion are stored on file in such a way that the subsequent off-line analysis can be done easily using the Matlab software package. Data are also displayed for viewing.

b. A counter is updated at each sampling time and the content is stored as a means to check whether or not any data is lost.

c. The operator can turn the compressor ON or OFF at will.

d. The operator can put two control loops under automatic control or he can 'open' the loops for controller tuning. In order to obtain the open-loop output response to a step change in the valve position, the motorized valve can be activated from the keyboard.

e. The status of the DAS, the controllers, the compressor and the valves is constantly monitored and displayed.

f. The operator can disable the DAS and return to the main menu. The date and times at which the experiment starts and stops are displayed on the screen and also recorded on file.

3.6.1 Starting the DAS

Before enabling the DAS for A/D conversion, the operator has to

enter the sampling periods, T_{s1} and T_{s2} in this order; otherwise the default sampling periods will be used. The *starter*() routine is called when the <F1> key is pressed. Counter 1 is now loaded with the count value corresponding to T_{s1} and the countdown begins. All hardware interrupt requests except those from the DAS (IRQ2) and the keyboard (IRQ1) are masked off. Priority is given to IRQ2 to avoid conflict should both IRQ1 and IRQ2 occur at the same time. The flow chart of the *starter()* routine is shown in Flow chart 3a.

After the DAS is enabled, the computer is ready to accept interrupts from either the DAS or the keyboard. To indicate new sampled data are available, the *handler()* routine sets *High_SR* flag to 1 at sampling time T_{s1}. At sampling time T_{s2}, both *change_flag* and *High_SR* are set to 1.

From the keyboard, the operator can turn the compressor ON/OFf, switch the automatic controllers ON/OFF or he can manipulate the motorized valves for open-loop controller tuning.

While waiting for interrupts, the *measurement_display()* routine performs other tasks as shown in Flow chart 4.

3.6.2 Stopping the DAS

The operator can disable the DAS either with the <F2> or <Ctrl-~> key; however, the latter will cause a return to main menu. Flow chart 3b shows the sequence of actions when the DAS is disabled. Once the DAS is disabled with the <F2> key, the operator can refresh the screen using the <F9> key.

Table 10 shows the flags associated with Display screen.

3.7 Help screen

The Help screen is implemented to provide the operator with an on-line operation manual. The display utilises video services provided by the BIOS interrupt 10hex.

The operator can now scroll the document using either the <Up/Down> arrow key or the <PgUp/PgDn> key. He can also go back to Display screen using the <Left> arrow key or to PID screen using <Right> arrow key. The <Left/Right> arrow key will reset *I_Oflag* to 1 to indicate exit from I_O() routine. Flow chart 5 shows the sequence of actions which occurs when one is at Help screen.

3.8 Evaluation of refrigerant thermodynamic properties

Thermodynamic properties of refrigerant R 22 can be evaluated using computer routines developed by Chan et al [24,25,26]. These routines are widely used because they offer great accuracy and cover a wide range of applications.

There are two main concerns which make the routines not suitable for the current research. First, extensive computation time is required which adds to real time overhead in the current interrupt-driven computer system. Secondly, language conversion is necessary because the routines are written in Fortran.

An alternative which seems more appropriate for the current research is proposed by Cleland [27]. In this method, curve-fitted equations are used instead of routines, which greatly improve computation speed; however, the trade-off is less accuracy and a limited range of applicability.

In going around the refrigerant loop (Fig.2.1), refrigerant thermodynamic properties can be calculated based upon [27].

a. Calculating liquid enthalpy h_3: Assuming that no pressure losses occur in the high side of the heat pump. T_{sat} corresponding to to discharge pressure P_H is given by

$$T_{sat} = \frac{a_2}{\ln(P_H) - a_1} - a_3 \qquad (3.1a)$$

where T_{sat} is in [°C] and P_H in [Pa]. The subcooling is hence:

$$\Delta T_b = T_{sat} - T_7 \qquad [°C] \qquad (3.1b)$$

When (3.1b) $\geq$ 0, h_3 is given by

$$h_3 = a_4 + a_5 T_7 + a_6 (T_7)^2 + a_7 (T_7)^3 \qquad [J/Kg] \qquad (3.1c)$$

b. Calculating enthalpy h_4: The ideal condition can be assumed where (1.1d) holds. Accordingly,

$$h_4 = h_3 \qquad [J/Kg] \qquad (3.2)$$

c. Superheated vapour enthalpy h_1:

$$T_2 = T_s \qquad [°C] \qquad (3.3a)$$

and

$$T_1 = T_{sat} \qquad (3.3b)$$

$$\text{Superheat} = \Delta T_s = T_s - T_{sat} \qquad (3.3c)$$

$$h_{i2} = h_{i1}[1 + a_{13}\Delta T_s + a_{14}(\Delta T_s)^2 + a_{15}(\Delta T_s)(T_{sat}) + a_{16}(\Delta T_s)^2(T_{sat}) + a_{17}(\Delta T_s)(T_{sat})^2 + a_{18}(\Delta T_s)^2(T_{sat})^2] \qquad (3.3d)$$

$$h_1 = h_{i2} + a_{12} \qquad [J/Kg] \qquad (3.3e)$$

d. Mass flow rate m_r: When no heat loss at the evaporator is assumed, (1.1a) can be used to calculate the mass flow rate of the

refrigerant.

$$m_r = \frac{Q_L}{h_1 - h_4} = \frac{m_{we} c_p (T_6 - T_8)}{h_1 - h_4} \quad [Kg/s] \qquad (3.4)$$

where h_1 and h_4 are given by (3.3e) and (3.1c) respectively. m_{we}, c_{we}, T_6 and T_8 are all known quantities.

e. Rate of heat removed from the condenser: The rate of heat removed from the condenser can be calculated from

$$\text{Rate of heat removed} = m_{wc} c_p (T_4 - T_3) \quad [J/Kg] \qquad (3.5)$$

f. Rate of heat transferred to the condenser Q_H: When W is known, Q_H can be calculated from (1.2).

The implementation of the above equations in the program remains to be done . The coefficients a_i's are shown in Table 11.

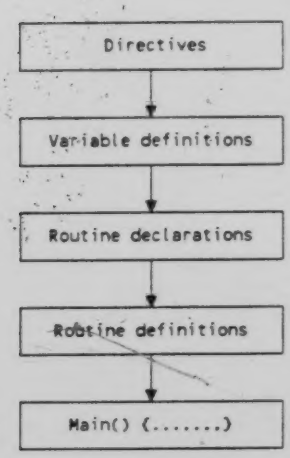


Fig.3.1: Program structure

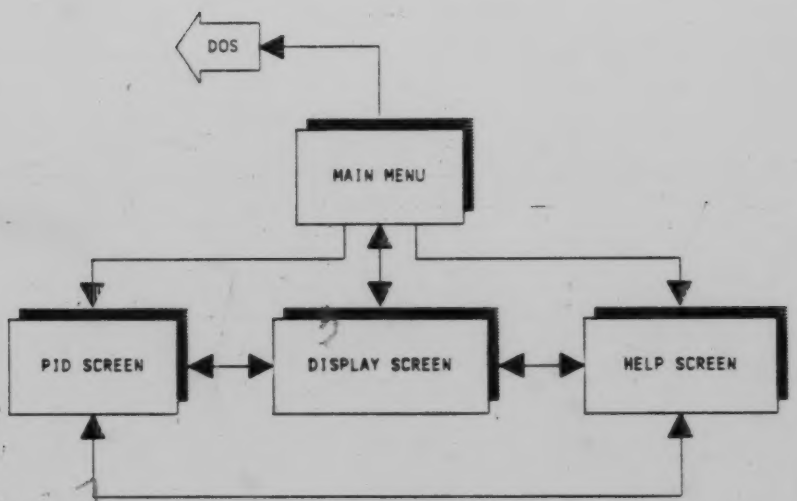


Fig.3.2: Block diagram of menu-driven program

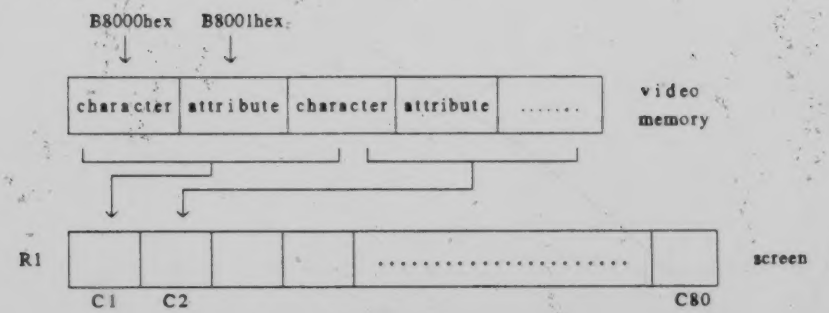


Fig.3.3: Video memory to screen mapping.

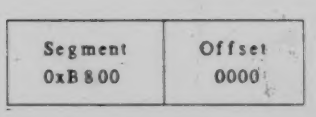


Fig.3.4: A far pointer to video memory.

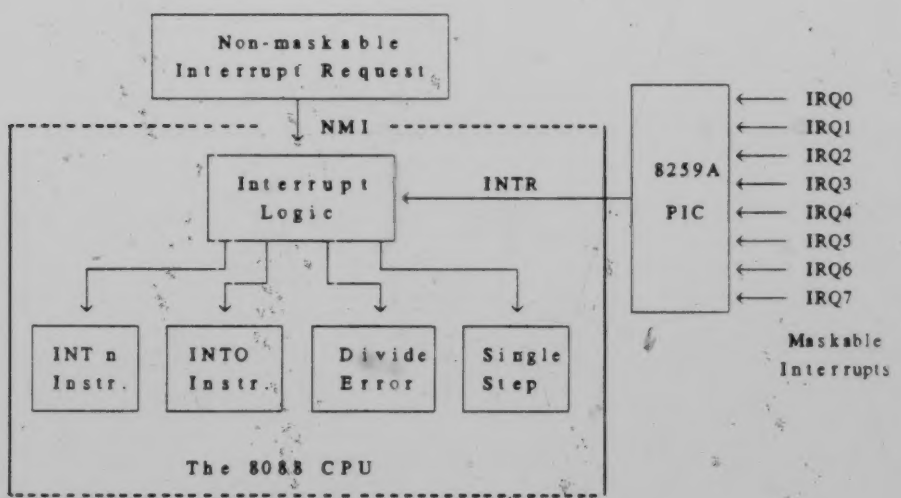


Fig.3.5: 8088 interrupt sources.

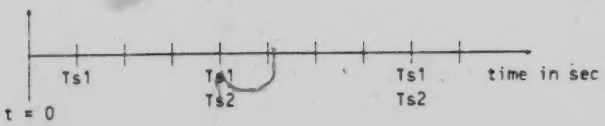


Fig.3.6: Relationship between Ts1 and Ts2

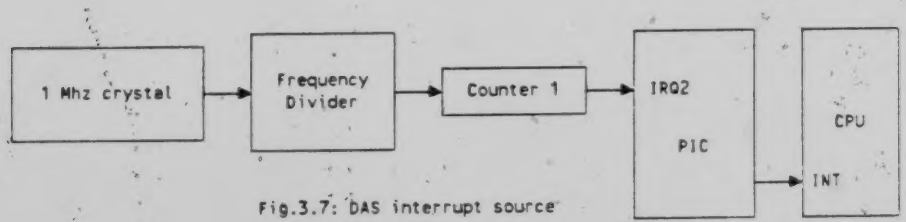


Fig.3.7: DAS interrupt source

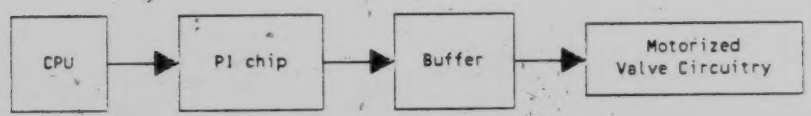


Fig.3.8: Arrangement of control signal to motorized valve

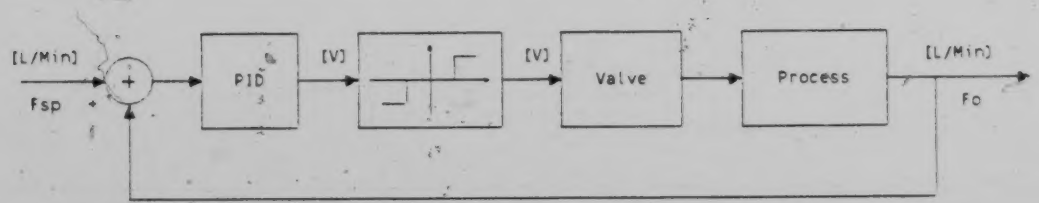


Fig.3.9: Block diagram of flow control loop

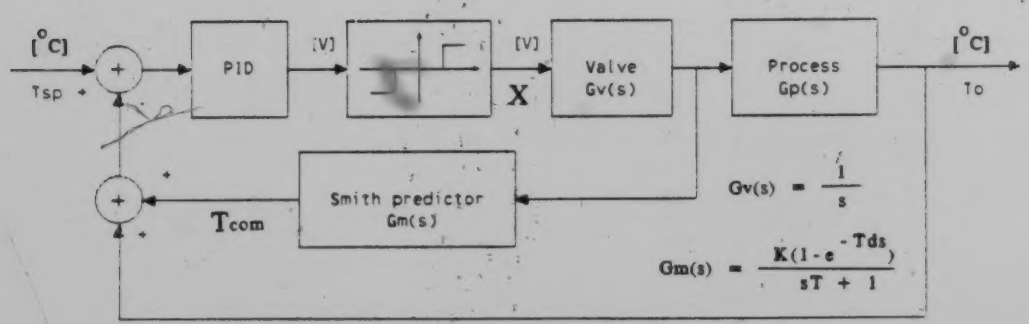


Fig.3.10: Block diagram of compensated temperature loop

SCREEN	KEY	FUNCTION
PID	F3	select flow controller for parameter update.
	F4	select temperature controller.
	↑ or ↓	select parameter to be updated.
DISPLAY	a	select valve 1 (hot water valve).
	b	open valve 1.
	c	close valve 1.
	d	select valve 2 (cold water valve)
	e	close valve 2.
	f	open valve 2.
	g	stop valve 1.
	h	stop valve 2.
	i	turn on compressor.
	j	turn off compressor.
	F1	enable the DAS.
	F2	disable the DAS.
	F5	turn on automatic controller.
	F6	turn off automatic controller.
	F9	refresh screen.
	Ctrl-~	return to main menu.
HELP	↑ or ↓	scroll 2 lines up/down.
	PgUp/PgDn	scroll page up/down.
	⟷	screen switching.

Table 2: Keyboard assignment.

SEQUENCE	EFFECT
ESC[#A	moves cursor up # rows without changing column. Defaut value for # is 1. Sequence is ignored if cursor is already on top row
ESC[#B	moves cursor down # rows without changing column. Default value for # is 1. Sequence is ignored if cursor is already on bottom row.
ESC[#C	moves cursor forward # columns on same row Default value for # is 1. Sequence is ignored if cursor is alereadv in rightmost column.
ESC[#D	moves cursor backward # columns on same row. Default value for # is 1.Sequence is ignored if cursor is already in leftmost column.
ESC[#;#H	moves cursor to position (#,#). Default value for # is 1. If no parameter is given cursor is moved to home position (0,0).
ESC[2J	clears screen and homes cursor.
ESC[K	erases from (and including) cursor position to end of line.

Table 3: Cursor control sequences.

INTERRUPTS	PRIORITY
Divide error, INT n, INTO	highest
NMI	
INTR	
Single-step	lowest

Table 4: 8088 Interrupt priorities.

LEVEL	TYPE (hex)	USAGE
NMI	02	Parity (highest priority)
IRQ0	08	Timer
IRQ1	09	Keyboard
IRQ2	0A	Unused
IRQ3	0B	Asynchronous communications (secondary) SDLC communications BSC (secondary)
IRQ4	0C	Asynchronous communications (primary) SDLC communications BSC (primary)
IRQ5	0D	Fixed disk
IRQ6	0E	Diskette
IRQ7	0F	Printer (lowest priority)

Table 5: 8088 hardware interrupt listing.

PORT ADDRESS (Hex)	COMMAND WORD
20	ICW1
21	ICW2
21	ICW3
21	ICW4
21	OCW1
20	OCW2
20	OCW3

Table 6: Addressing structure of the PIC.

FREQUENCY F	SAMPLING PERIOD Ts
F1 (1MHz)	0 to 0.064 sec
F2 (100KHz)	0.065 to 0.649 sec
F3 (10KHz)	0.65 to 6.49 sec
F4 (1KHz)	6.5 to 64.9 sec
F5 (100Hz)	65 to 649 sec

Table 7: Frequency sources for DAS counter 1.

A/D ANALOG INPUT (V)	A/D OUTPUT IN TWO'S COMPLEMENT FORMAT		TWO'S COMPLEMENT TO VOLTAGE CONVERSION
	HEX	DECIMAL	
10	07FF	2047	2047/204.8 ≃ 9.9951172
.	.	.	.
.	.	.	.
.	.	.	.
.	0001	1	1/204.8 ≃ 0.0048828
0	0000	0	0/204.8 = 0
.	FFFF	65535	(65535-65536)/204.8 ≃ -0.0048828
.	.	.	.
.	.	.	.
.	.	.	.
-10	F800	63488	(63488-65536)/204.8 = -10

Table 8: Input/output mapping of the A/D converter.

KEY	FLAGS	SET	STATUS DISPLAY
F3	hit_F3	1	(highlight Kp1
	hit_F4 (reset if == 1)	0	initially)
	control_select, tuningF	1	
F4	hit_F4	1	(highlight Kp2
	hit_F3 (reset if == 1)	0	initially)
	control_select	-1	
	tuningT	1	
↑ or ↓			(highlight para meter to be updated)
←	I_Oflag	1	(exit I_O() and switch to Help screen)
→	I_Oflag	1	(exit I_O() and switch to Disp lay screen)

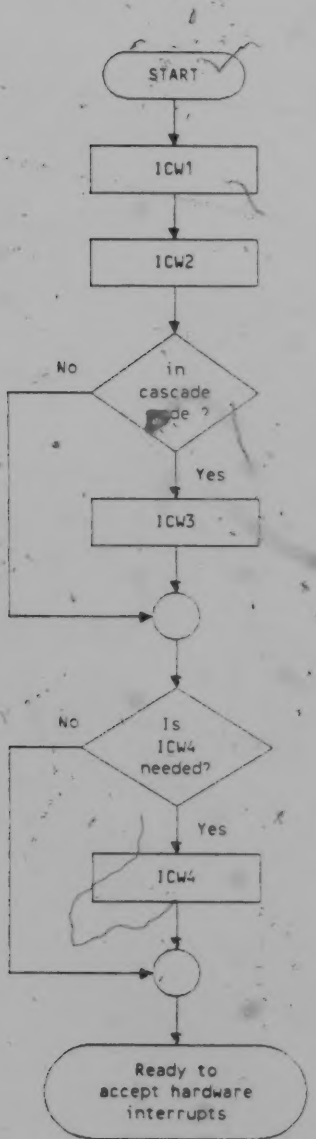
Table 9: Flags associated with PID screen.

KEY	FLAGS	SET	STATUS DISPLAY
a	valve_flag1	1	
b	valve_dir1, hit_key1	1	VALVE1: OPEN
	stall_flag1	0	
c	valve_dir1, hit_key1	-1	VALVE1: CLOSE
	stall_flag1	0	
d	valve_flag2	1	
e	valve_dir2, hit_key2	-1	VALVE2: CLOSE
	stall_flag2	0	
f	valve_dir2, hit_key2	1	VALVE2: OPEN
	stall_flag2	0	
g	valve_flag1, valve_dir1	0	VALVE1: STOP
	hit_key1	2	
h	valve_flag2, valve_dir1	0	VALVE2: STOP
	hit_key2	2	
i	Compressor_on, hit_key3	1	COMPRESSOR: RUNNING
j	Compressor_off	0	COMPRESSOR: STOP
	hit_key3	1	
F1	running_flag	1	DAS: RUNNING (Date and time are are displayed)
F2	running_flag	0	DAS: STOP
	valve_flag1, valve_flag2	0	(stop time is
	valve_dir1, vulve_dir2	0	displayed)
F5	hit_autocontrol, auto_control	1	AUTOCONTROL:ON(WL&TL) or :ON (WL) or :ON (TL) or :N/A depending on TuningF and TuningT
F6	hit_autocontrol, auto_control	-1	AUTOCONTROL: OFF
	tuningF, tuningT	0	
F9			DAS: STANDBY (Start and stop times are cleared)
Ctrl-~	stop	1	(return to main menu)
	running_flag	0	
	valve_flag1, valve_flag2	0	
	valve_dir1, valve_dir2	0	
←	I_Oflag	1	(exit I_O() routine and switch to PID screen)
→	I_Oflag	1	(exit I_O() routine and switch to Display screen)

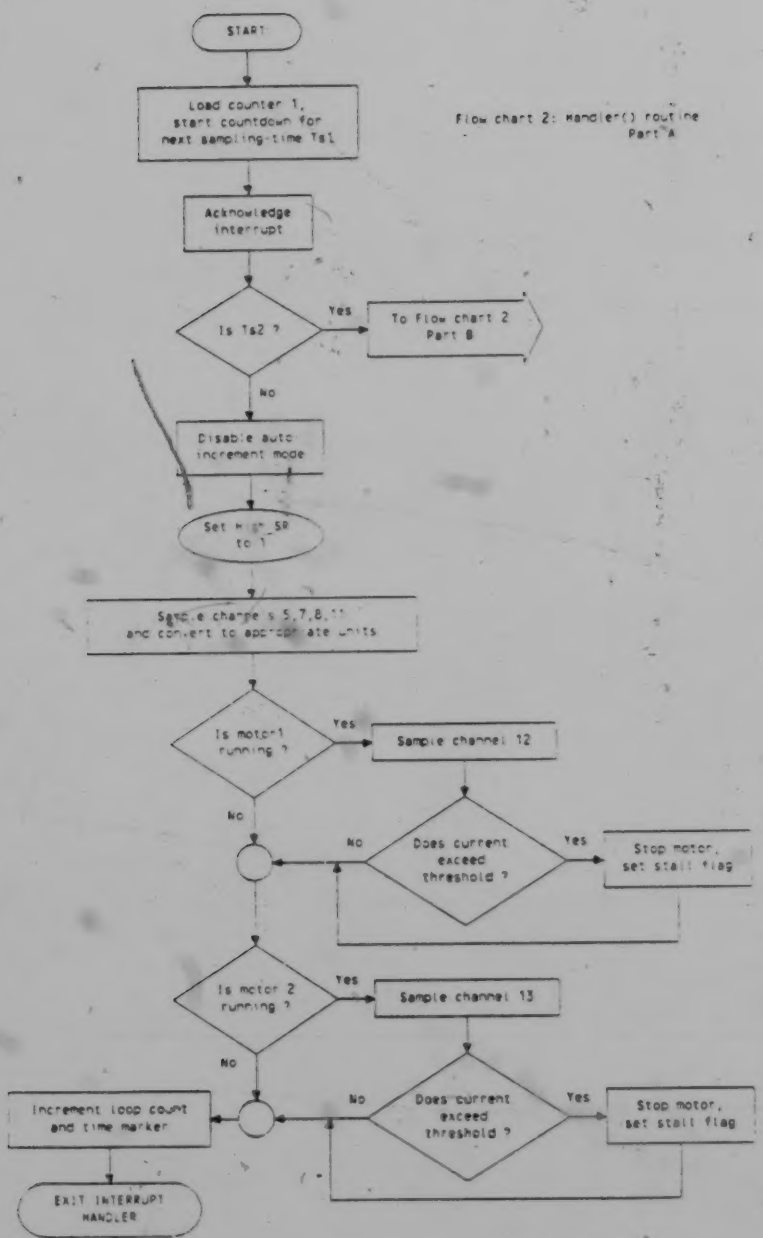
Table 10: Flags associated with Display screen.

COEFFICIENT	VALUE
a_1	21.25384
a_2	-2025.4518
a_3	248.94
a_4	-200000
a_5	1170.36
a_6	1.68674
a_7	5.2703×10^{-3}
a_{12}	155482
a_{13}	2.85446×10^{-3}
a_{14}	4.0129×10^{-7}
a_{15}	13.3612×10^{-6}
a_{16}	-8.11617×10^{-8}
a_{17}	14.1194×10^{-8}
a_{18}	-9.53294×10^{-10}

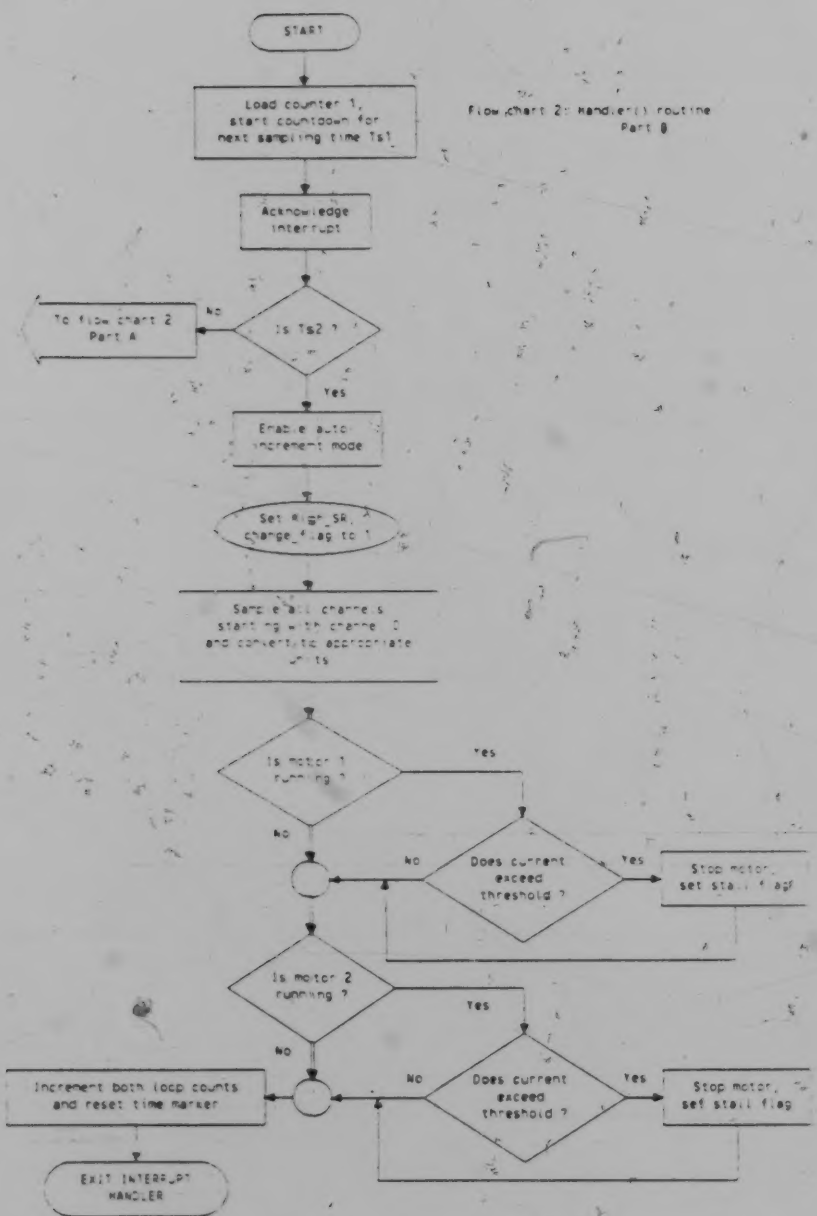
Table 11: Coefficients to evaluate refrigerant thermodynamic properties.

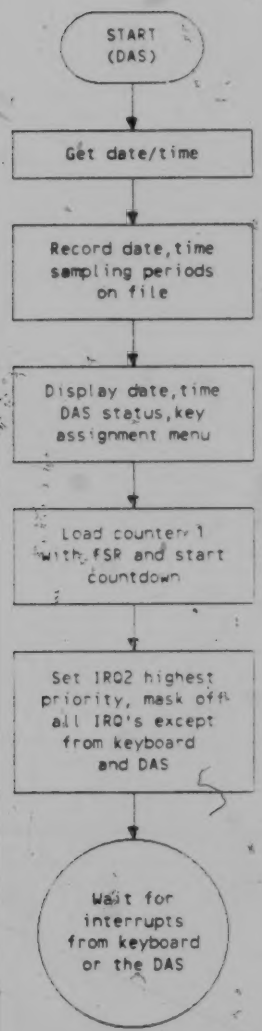


Flow chart 1: PIC initialization

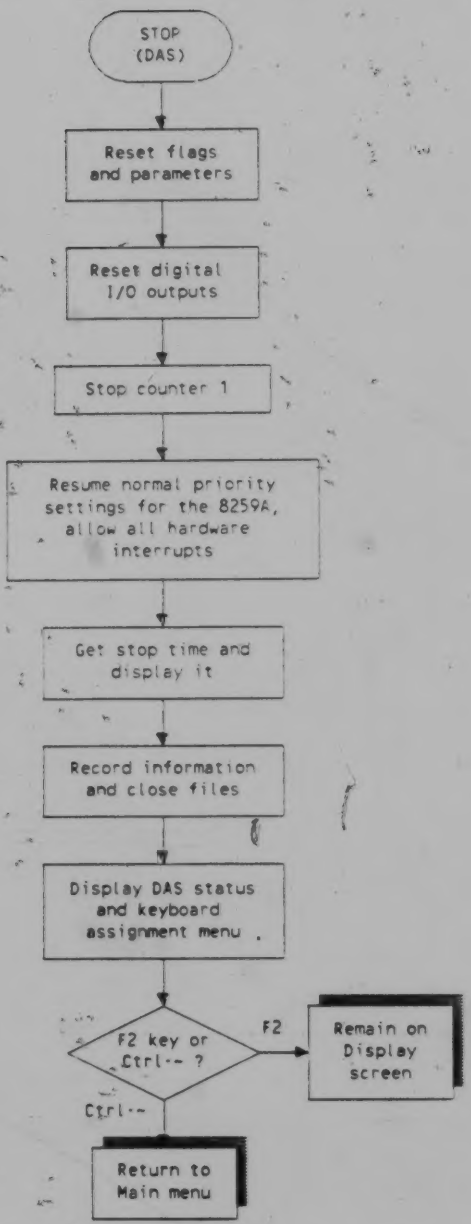


Flow chart 2: Handler() routine
Part A

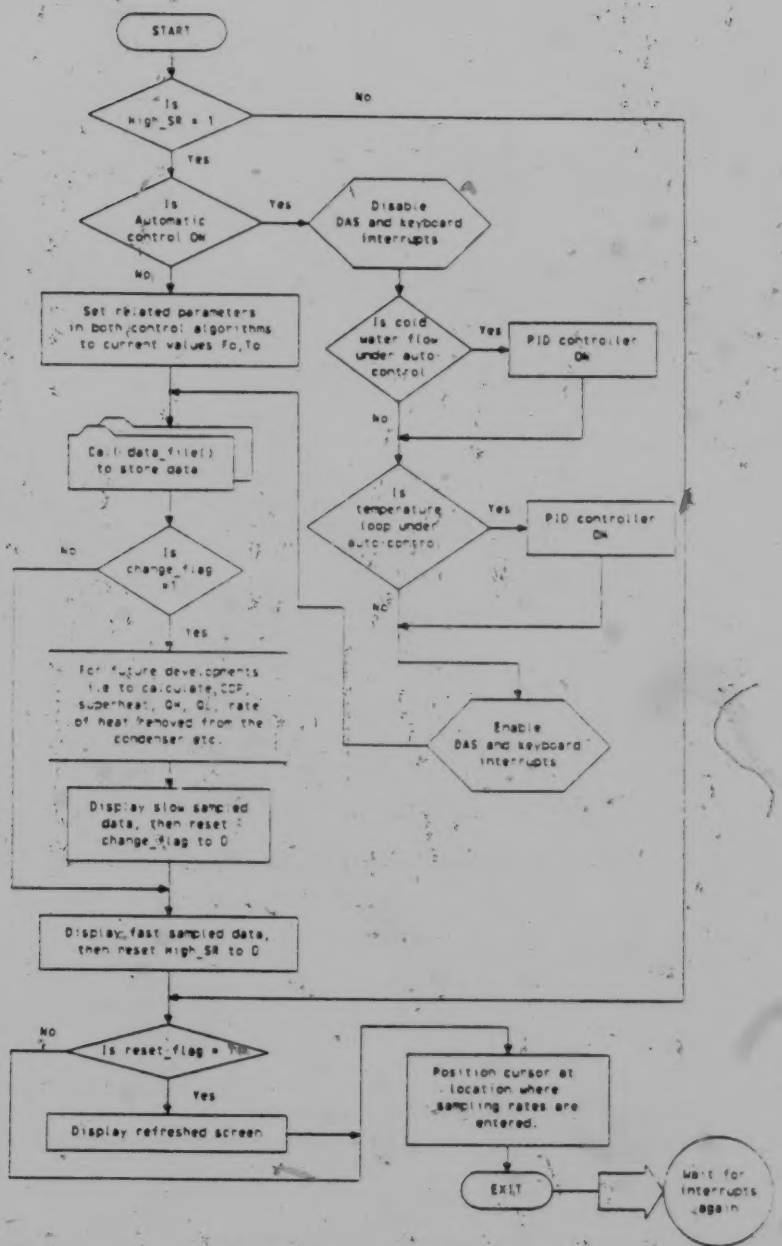

START
Load counter 1, start countdown for next sampling time Ts1
Flow chart 2: Handler() routine Part B
Acknowledge interrupt
Is Ts2 ?
No
To flow chart 2 Part A
Yes
Enable auto increment mode
Set High_SR change_flag to 1
Sample all channels starting with channel 0 and convert to appropriate units
Is motor 1 running ?
Yes
No
Does current exceed threshold ?
No
Yes
Stop motor, set stall flag
Is motor 2 running ?
Yes
No
Does current exceed threshold ?
No
Yes
Stop motor, set stall flag
Increment both loop counts and reset time marker
EXIT INTERRUPT HANDLER



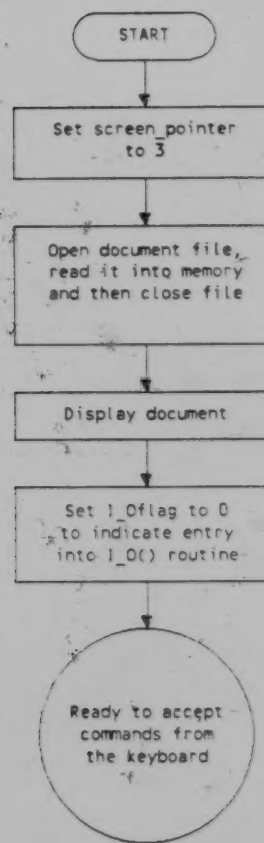
Flow chart 3a: Starter() routine.



Flow chart 3b: Stopper() routine.



Flow chart 4: Measurement_display() routine


START
Set screen_pointer to 3
Open document file, read it into memory and then close file
Display document
Set I_Oflag to 0 to indicate entry into I_O() routine
Ready to accept commands from the keyboard

Flow chart 5: Help screen

CHAPTER 4
CONTROLLER TUNING

There are two controlled variables in the heat pump system: one is the cold water flow rate into the condenser (F_o) and the other the water temperature to the evaporator (T_o). In either case, the final control element is a motorized valve connected to the power supply via a switching relay whose characteristics are shown in Fig.4.

One can identify three cases:

1. $-\Delta \leq c \leq \Delta$: No control action
2. $c > \Delta$: The valve is being opened.
3. $c < -\Delta$: The valve is being closed.

The dead zone Δ is incorporated into the relay characteristics in order to prevent frequent on-off switching, or chattering, which might shorten the life of the relay. Unfortunately, Δ will produce steady state offsets unless PID control is used, and its choice will greatly influence the controller's settings.

The control objectives are to ensure stability of the process, rejection of disturbances, and minimization of the steady state error in the controlled output.

Various conventional tuning techniques can be found in the literature. They are generally based on either the open-loop or the closed-loop response. Many processes can be modeled as the first-order lag plus dead time, whose characteristics can be found with the open-loop method. The Process Reaction curve yields information on the process gain, time constant, and dead time, from which the

controller's settings are calculated. Miller et al [28] compare various tuning techniques based on this method. In the closed-loop or Ultimate Sensitivity method proposed by Ziegler and Nichols [29], the controller's settings are derived from the critical gain K_P at which the controlled variable undergoes sustained oscillations, and the corresponding period of oscillation; however care should be taken because this method does not work on processes with dead time, nor can it apply in the case where cycling conditions are not permitted [30].

4.1 Controlling the water flow rate into the condenser (F_o)

Fig.4.1a shows the cold water flow control loop and Fig.3.9 its corresponding block diagram. The dynamics of the measuring sensor is ignored because it is insignificant when compared to the process dynamics.

4.1.1 Tuning the controller

Since the flow responds almost instantaneously to a change in the valve position, the Ultimate Sensitivity method is used to tune the controller.

Δ is arbitrarily chosen to be 0.2V. After the loop is 'closed', and for a given set point change, the proportional gain K_P is increased until sustained oscillations in the controlled output are observed. It is found that $K_{P_{critical}} = 0.18$ V/(L/Min) and $T_{critical} =$ 24.2 sec. As a rule of thumb for a P controller:

$$K_P = 0.5 * K_{P_{critical}} = 0.09 \text{V/(L/Min)}$$

The P and PD controllers are unsatisfactory due to large offsets in the output response. The maximum steady state error for the case of the P controller is found to be

$$|E_{ss}|_{max} = \frac{\Delta}{K_p} = \frac{0.2 \quad [V]}{0.09 \ [V/(L/Min)]} \simeq 2.22 \ L/Min$$

The PID controller is therefore employed to eliminate the offset. As seen later on, the integral action tends to destabilize the system. In this regard, a compromise is made by allowing some degree of tolerance in the steady state error. Within this allowed range, the integral mode is turned off.

The tuning criteria are based upon performance characteristics in terms of small overshoots, short rise time and setting time. Table 12 compares the controller's settings obtained from further fine tuning to those proposed by Ziegler and Nichols.

From Table 12, it is seen that the integral action is negligible when compared to the others. Its presence, however, guarantees that the steady state error is within the predefined $\mp$ 0.5L/Min tolerance. The choice of the K_i term is affected by a destabilizing effect of an integration already embedded in the system, attributed to by the behavior of the final control element i.e a change in the valve position depends upon the duration of power input to the valve. Increasing K_i may cause oscillations and lead to instability. The contribution of the K_d term is the most important because it not only introduces a stabilizing effect to the system but also makes the controller more robust.

4.1.2 Experimental results

For K_p = 0.09 [V/(L/Min)], K_d = 0.1635 [V.s/(L/Min)], K_i = 0.0045 [V/(L.s/Min)], Δ = 0.2 [V], and ∓ 0.5 L/Min tolerance, the PID controller is tested at various changes in the set point. Fig.4.1.1 to 4.1.4 show the output responses . The impact of disturbances on system performance is also studied and the corresponding output responses are shown in Fig.4.1.7 to 4.1.10, where the tolerance is set at ∓ 1.0L/Min and the controller's parameters remain unchanged.

In general, the PID controller performs satisfactorily. In most cases, it is observed that there are no oscillations in the transient period and that the output response achieves the desired set point in a relatively short time, and with acceptable control effort i.e short durations of the input pulse to the valve (D in the plots).

Fig.4.1.2 particularly shows the effect of integral action on minimizing the steady state error. It also underlines a need for compromises between conflicting objectives in process control. In this plot, the output response quickly settles down to a steady state value whose offset is outside the ∓ 0.5 L/Min tolerance. Consequently, the integral action remains active and becomes the dominant control mode. Since K_i is of a small magnitude, a long 'wait' time elapses before any corrective action can take place. The output response then undergoes a transient period before eventually approaching the desired set point. This 'wait' time can be reduced by increasing K_i, which in turn may cause the transient period to become more oscillatory and even destabilize the process. If, on the other hand, a tolerance of ∓ 1.0 L/Min were allowed, both the transient period and a long settling

time could have been avoided, thus improving the behavior of the output response. Since the acceptable tolerance is specified by the user, the controller needs to be retuned if the tolerance band is changed so that the controller's performance remains satisfactory.

Disturbances or loads are always present in the system. They are unpredictable and adversely affect the process output. The specified tolerance accounts for these undesirable effects as long as deviations from the set point are within the allowed range; otherwise, the PID controller will carry out corrective action to suppress them. Fig.4.1.7 to 4.1.10 show the output responses in the presence of the disturbance. In these cases, after the output response achieves its desired steady state value, the disturbance is deliberately introduced to the process by reducing the water supply pressure. Due to a large derivative mode, corrective action immediately takes place to maintain the output at its desired level. When the disturbance is removed, the PID controller again reacts promptly to make the output return to its setpoint.

4.2 Controlling the evaporator inlet water temperature (T_o)

Fig.4.2a shows the temperature control loop and Fig.3.10 its corresponding block diagram. The manipulated variable in this case is F_h and the disturbances are T_h, T_c, and F_c. The dynamics of the temperature sensor and the flow regulator are both neglected.

4.2.1 Dead time compensation

As seen in a typical open-loop output response curve (Fig.4.2b),

there is a large dead time in the temperature loop due to thermal inertia, transport delay etc.. Dead time presents a real challange in process control because it significantly contributes to process instability. As a result, the Ultimate Sensitivity method can not be applied in tuning the temperature controller. One of the techniques to compensate for dead time is the Smith predictor [23]. Fig.3.10 shows the block diagram of the compensated temperature loop, where the valve is modeled as a pure integrator and the process a first-order lag plus dead time. The process model so chosen is justified by examining Fig.4.2b.

In Fig.3.10, the transfer functions of the valve, the process, and the Smith predictor are given by

$$G_v(s) = \frac{1}{s}\ ,\ G_p(s) = \frac{K_1 e^{-T_{d1}s}}{T_1 s + 1},\ G_m(s) = \frac{K(1 - e^{-T_d s})}{Ts + 1}$$

Process gain K, time constant T, and dead time T_d need to be known for the implementation of the Smith predictor. They are experimentally obtained from the open-loop output response curves. It is observed that K varies from one operating point to the next, but T and T_d remain almost the same (Table 13). The different values for K reflect the physical constraints upon the temperature of the mixture:

$$T_c = T_{omin} \leq T_o \leq T_{omax} < T_h \qquad (4.1)$$

It is seen that when the valve is completely closed, T_o equals the temperature of the cold water source T_c. When the valve is completely opened, T_o saturates to a maximum value, but it never exceeds T_h, the temperature of the hot water source, unless the cold

water source is removed.

a. Effectiveness of the Smith predictor: From Fig.3.10, one obtains the overall loop transfer function

$$G_{eff}(s) = G_v(s)\left[G_p(s) + G_m(s)\right]$$

$$= \frac{1}{s}\left[\frac{K_1 e^{-T_{d1}s}}{T_1 s + 1} + \frac{K(1 - e^{-T_d s})}{Ts + 1}\right] \quad (4.2)$$

In the ideal case, i.e $K_1 = K$, $T_1 = T$, and $T_{d1} = T_d$, (4.2) becomes

$$G_{eff}(s) = \frac{K}{s(Ts + 1)} \quad (4.3)$$

Unfortunately, perfect compensation can never be achieved due to unavoidable modeling errors, and they will greatly influence the performance of the compensator.

b. Implementation of the Smith predictor: With the assumed perfect compensation, from Fig.3.10 one has

$$\frac{T_{com}(s)}{X(s)} = G_v(s)G_m(s)$$

$$= \frac{K(1 - e^{-T_d s})}{s(Ts + 1)} \quad (4.4)$$

or $$(s^2T)T_{com} + sT_{com} = KX - KXe^{-T_d s} \quad (4.5)$$

In the time domain (4.5) becomes

$$T\frac{d^2T_{com}}{dt^2} + \frac{dT_{com}}{dt} = KX(t) - KX(t - T_d) \quad (4.6)$$

(4.6) is then converted to the discrete form

$$T\frac{T_{com}(n) - 2T_{com}(n-1) + T_{com}(n-2)}{(T_{s1})^2} + \frac{T_{com}(n) - T_{com}(n-1)}{T_{s1}} = KX(n) - KX(n - n_{T_d})$$

or

$$T_{com}(n) = \frac{KT_{s1}^2[X(n)-X(n-n_{Td})]+(2T+T_{s1})T_{com}(n-1)-TT_{com}(n-2)}{T + T_{s1}} \quad (4.7)$$

In (4.7), X is the valve input which can assume one of the three voltage levels ±90V and 0V. The sampling rate T_{s1}, setpoint T_{sp} as well as process characteristics K, T, and T_d are entered from the keyboard. The error signal to the controller is therefore given by

$$E(n) = T_{sp} - [T_o(n) + T_{com}(n)] \quad (4.8)$$

Alternatively, one can rewrite (4.4) as

$$\frac{T_{com}(s)}{X(s)} = \frac{K}{s(Ts + 1)} - \frac{Ke^{-T_d s}}{s(Ts + 1)}$$

or

$$T_{com}(s) = \frac{KX(s)}{s(Ts + 1)} - \frac{Ke^{-T_d s}X(s)}{s(Ts + 1)}$$

$$= T_a(s) - T_b(s) \quad (4.9)$$

T_b and T_a represent the model output with and without dead time, respectively. In the discrete time domain, they are described by

$$T_a(n) = \frac{K(T_{s1})^2X(n) + (2T + T_{s1})T_a(n-1) - TT_a(n-2)}{T + T_{s1}} \quad (4.10)$$

$$T_b(n) = \frac{K(T_{s1})^2X(n-n_{Td}) + (2T + T_{s1})T_b(n-1) - TT_b(n-2)}{T + T_{s1}} \quad (4.11)$$

When (4.10) and (4.11) are implemented in the computer program, the constraints in (4.1) have to be included. The error signal now becomes

$$E(n) = T_{sp} - [T_o(n) + T_a(n) - T_b(n)] \quad (4.12)$$

With (4.7) being replaced by (4.10) and (4.11), one can easily verify the accuracy of the process model.

4.2.2 Describing function method

Tuning the compensated temperature loop involves the selection of controller parameters such that the control objectives are met. Despite the implementation of the Smith predictor in the loop, stability is still of primary concern. Since the process characteristics are experimentally known, stability of the loop can be investigated using the Describing function technique. Unlike the phase-plane method, the technique does not reveal the time-domain behavior of the output except information on stability from a frequency-domain point of view. This technique amounts to an examination of whether sustained oscillations at some frequency and amplitude can be maintained in the loop. In the analysis, the nonlinear element is replaced by the best linear gain approximation and only the fundamental harmonic component of its output is considered. The higher harmonics are ignored due to the fact that most process control sytems exhibit a low-pass filtering effect which can substantially attenuate the higher harmonics other than the fundamental [31]. Fig.4.3 shows the block diagram of a nonlinear feedback control loop, where the forward path is broken at either end of the nonlinear element.

Suppose that a sinusoidal signal is present at the output of the nonlinear element

$$E_o(t) = E_o \sin\omega t \qquad (4.13)$$

In going through the loop and with $R_{sp} = 0$, the signal undergoes a phase and gain change before being fed back to the nonlinear element

$$E_i(t) = g(\omega)E_o\sin[\omega t + \theta(\omega)] \qquad (4.14)$$

where

$$g(\omega) = |G(j\omega)|$$

$$= |G_c(j\omega)G_v(j\omega)[G_p(j\omega) + G_m(j\omega)]| \qquad (4.15)$$

and $$\theta(\omega) = \angle\, G(j\omega)$$

$$= \angle\, G_c(j\omega) + \angle\, G_v(j\omega) + \angle \left\{G_p(j\omega) + G_m(j\omega)\right\}$$

The nonlinear element is represented by its describing function

$$N(x) = M(x)e^{j\Psi(x)} \qquad (4.16)$$

where its gain and phase depend on the amplitude x of the input signal $E_i(t)$. The output signal now becomes

$$E_o(t) = - M(x)g(\omega)\sin[\omega t + \theta(\omega) + \Psi(x)]$$

$$= M(x)g(\omega)\sin[\omega t + \pi + \theta(\omega) + \psi(x)] \qquad (4.17)$$

The limit cycle exists when (4.13) equals (4.17). In other words, the following conditions must be satisfied

$$\begin{cases} M(x)g(\omega) = 1 & (4.18a) \\ \theta(\omega) + \psi(x) = (2n + 1)\pi; \quad n \text{ is an integer} & (4.18b) \end{cases}$$

Combining (4.18a) and (4.18b) yields

$$G(j\omega) = - \frac{1}{N(x)} \qquad (4.19)$$

Any intersection between $G(j\omega)$ and $-1/N(x)$ loci on the Nyquist diagram will be the solution of (4.19), indicating the existence of a stable limit cycle. The amplitude and frequency of oscillation can be found at the point of intersection. On the other hand, when no intersection occurs, the stability of the system can be assessed using

the normal Nyquist criterion with respect to any point on the -1/N(x) locus.

a. Stability analysis of the compensated temperature loop: For the compensated temperature loop shown in Fig.3.10, the describing function of the relay with dead zone is given by

$$N(x) = \frac{4M}{\pi x}\sqrt{1 - (\Delta/x)^2} \qquad (4.20)$$

or

$$-\frac{1}{N(x)} = -\frac{\pi x}{4M\sqrt{1 - (\Delta/x)^2}} \qquad (4.21)$$

In (4.21) $x > \Delta$ is the input amplitude, M = 90V, and Δ = 2V. -1/N(x) is a negative real number which has a maximum value of $-\frac{\pi\Delta}{2M}$ at $x = \Delta\sqrt{2}$.

From (4.15) the effective transfer function is given by

$$G(j\omega) = G_c(j\omega)G_v(j\omega)\left[G_p(j\omega) + G_m(j\omega)\right] \qquad (4.22)$$

$$= \left[K_p + \frac{K_i}{j\omega} + j\omega K_d\right]\frac{1}{j\omega}\left[\frac{K_1 e^{-T_d 1 j\omega}}{j\omega T_1 + 1} + \frac{(1 - e^{-T_d j\omega})K}{j\omega T + 1}\right] \qquad (4.23)$$

When there is perfect compensation, (4.23) becomes

$$G(j\omega) = \left[K_p + \frac{K_i}{j\omega} + j\omega K_d\right]\frac{1}{j\omega}\left[\frac{K e^{-T_d j\omega}}{j\omega T + 1} + \frac{(1 - e^{-T_d j\omega})K}{j\omega T + 1}\right]$$

$$= \left[K_p + \frac{K_i}{j\omega} + j\omega K_d\right]\frac{K}{j\omega(j\omega T + 1)} \qquad (4.24)$$

With K, T and T_d being known from Table 13 and for different values of K_p, K_d and K_i, the stability of the compensated temperature loop under PID control can be studied through simulation by plotting

either (4.24) or (4.23) and (4.21) on the Nyquist diagram and using the Nyquist criteria to determine stability.

b. Simulation: The effects of modeling errors on the Smith predictor in terms of process stability are studied through simulation; however, no rigorous analyses are attempted. An operating point (T_{sp} = 16°C) is selected at which the corresponding characteristics shown in Table 13 are assumed to represent the true process model. Deviations from these parameters are then introduced and their effects on process stability are studied with the aid of the Describing function method . The simulation program is written using the Matlab software package.

The Nyquist diagrams for both uncompensated and perfectly compensated temperature loops under proportional control (K_p = 1) are shown in Fig.4.3.1a, and the corresponding output responses are plotted in Fig.4.3.1b. In the uncompensated case, the closed -loop output response oscillates with a constant amplitude about the operating point (Fig.4.3.1b, curve A). The existence of a stable limit cycle is expected because of the intersection between $G_p(j\omega)$ and -1/N(x) on the Nyquist diagram (Fig.4.3.1a, solid curve). When there is perfect compensation, $G(j\omega)$ lies entirely on the third quadrant and neither does it intersect nor enclose the -1/N(x) locus (Fig.4.3.1a, dotted curve), indicating that the closed-loop output response is stable (Fig.4.3.1b, curve B).

Table 14 summarizes the effects of modeling errors on the performance of the Smith predictor. In the simulation, the physical

constraints imposed upon the controlled variables are T_c = 7°C and T_{omax} = 25°C. At the operating point of 16°C, the true process model is assumed to have the following characteristics:

Gain K = 0.0120 °C/(V.s), time constant T = 30s, dead time T_d = 14s

From the simulation, it is observed that the underestimated gain and overestimated time constant both adversely affect the compensator, and that the error in estimating dead time can be tolerable within a certain range only. In some cases, opposing effects cancel one another, leaving the compensator intact. Above all, the larger the modeling errors, the poorer the performance of the compensator.

4.2.3 Tuning the temperature controller

With the known process characteristics (Table 13), the design of the PID controller is based upon stability analysis using the Describing function method.

In the simulation, the process characteristics at the operating point 14°C is selected. Perfect compensation is assumed, and the effective transfer function $G(j\omega)$ is given by (4.24).

Fig.4.4.1 shows the effect of proportional control alone. The perfectly compensated temperature loop (K_P = 1) is always stable because the $G(j\omega)$ locus neither intersects nor encloses the -1/N(x) locus (solid curve). Increasing K_P moves the two loci closer to one another. As a result, the steady state error in T_o is reduced and the sensitivity increases. The price to pay for large K_P's is stability. The closer to the -1/N(x) locus, the more unstable the loop.

Under pure integral control, the temperature loop is always

unstable no matter how small K_i is chosen. This is because the $G(j\omega)$ locus lies entirely on the second quadrant and encloses the -1/N(x) locus as seen in Fig.4.4.2.

Fig.4.4.3 shows the effect of PD control. With K_p fixed at 2 (P control, solid curve), the addition of derivative mode increases stability but it does not improve the steady state error. Increasing K_d makes the $G(j\omega)$ and -1/N(x) loci further and further apart. Large K_d's are undesirable because of unnecessary control action when the process is susceptible to low-level disturbances.

The solid curve in Fig.4.4.4 corresponds to the case when the temperature loop is under PD control, with K_p and K_d being set at 2 and 3 respectively. The introduction of integral mode destabilizes the loop by moving the $G(j\omega)$ locus towards the -1/N(x) locus. Increasing K_i large enough i.e K_i = 0.09 will cause both loci intersect each other, resulting in the existence of a stable limit cycle.

Based upon the simulation results, the controller parameters are selected as shown in Table 15.

4.2.4 Experimental results

Fig.4.5.1 to 4.5.5 inclusively show the closed-loop output responses under proportional control. For $K_p \leq 3$ [V/ °C], the output responses are always stable; however, the steady state error can never be eliminated. Increasing K_p further will result in process instability as confirmed in Fig.4.5.5 for the case of K_p = 3.5 [V/ °C]. This is consistent with the simulation results that large K_p's tend to destabilize the system because the $G(j\omega)$ and -1/N(x) loci are closer

to each other.

The closed-loop output responses under PID control are shown in Fig 4.5.6 to 4.5.9 inclusively. It is seen that the steady state error is greatly reduced, well within the specified tolerance of ∓ 0.5°C. In particular, the effect of increasing K_I is shown in Fig.4.5.9, where a limit cycle exists for K_I = 0.05 [V/(°C.s)]. Again, the experimental result agrees with the fact that increasing Ki will pull the $G(j\omega)$ locus toward the -1/N(x) locus until an intersection occurs between the two loci, signifying the existence of a limit cycle.

The effects of disturbances upon the performance of the controller are studied by reducing either the hot or cold water supply pressure after the output response has reached its desired steady state. Fig.4.5.10 and 4.5.11 show the case when the effect of the disturbance is neglible; therefore no corrective action is necessary because the steady state error is still tolerable. Otherwise, as seen in Fig.4.5.12 to 4.5.16 inclusively, the controller will take immediate action to negate these undesirable effects.

It is also noted that the controlling input to the final control element is shown in every plot. The primary purpose is to show the duration of the input pulse to the valve.

With the settings specified specified in Table 15, the PID controller performs satisfactorily. Disturbances are suppressed and the steady state error is always kept within tolerance. Above all, stability of the process is guaranteed, thus vindicating the performance of the Smith predictor in the elimination of dead time in an otherwise uncontrollable temperature loop.

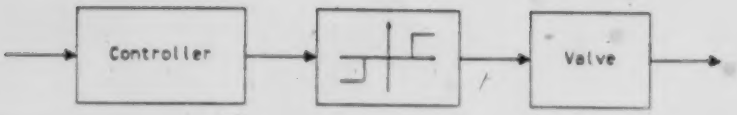


Fig.4: Relay characteristics

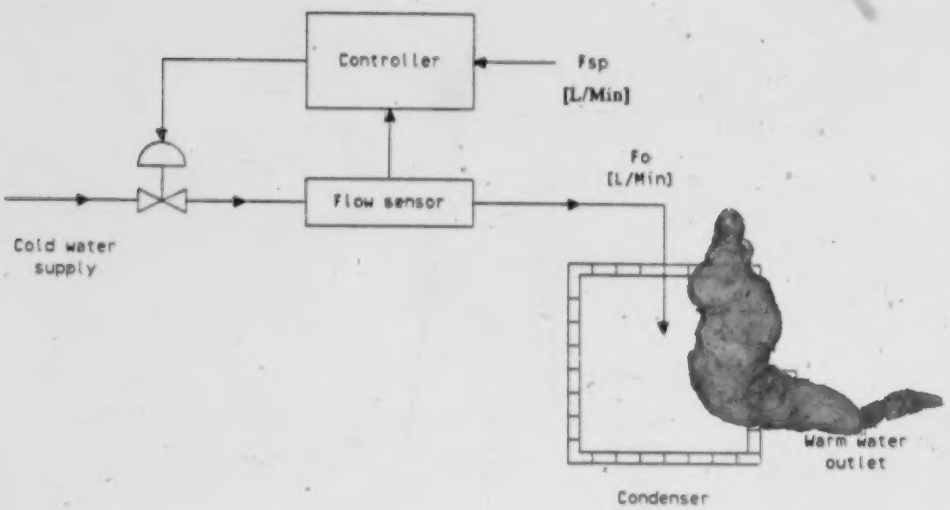


Fig.4.1a: Flow control loop

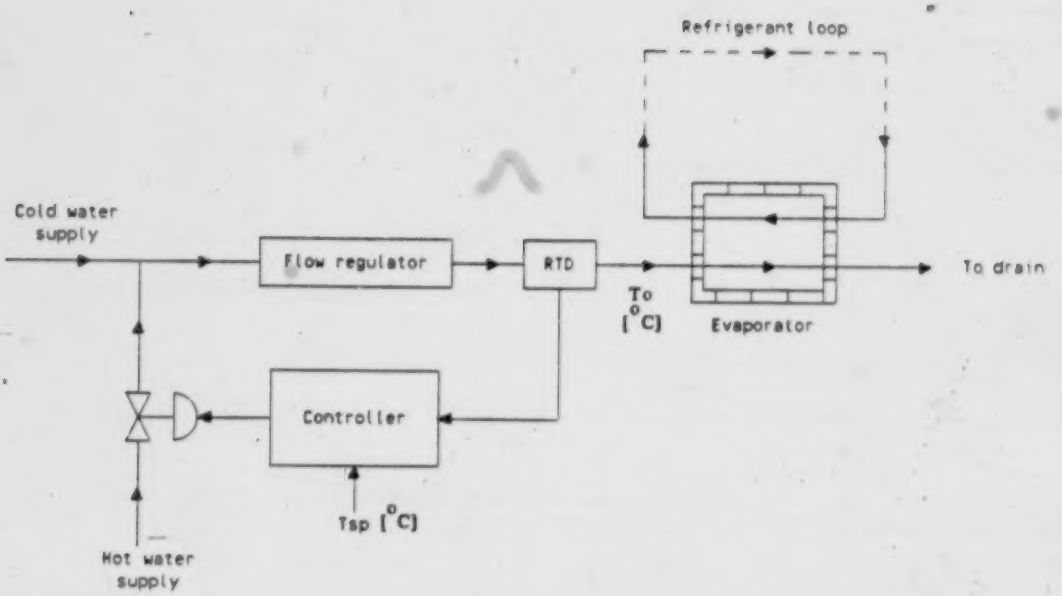


Fig.4.2a: Temperature control loop

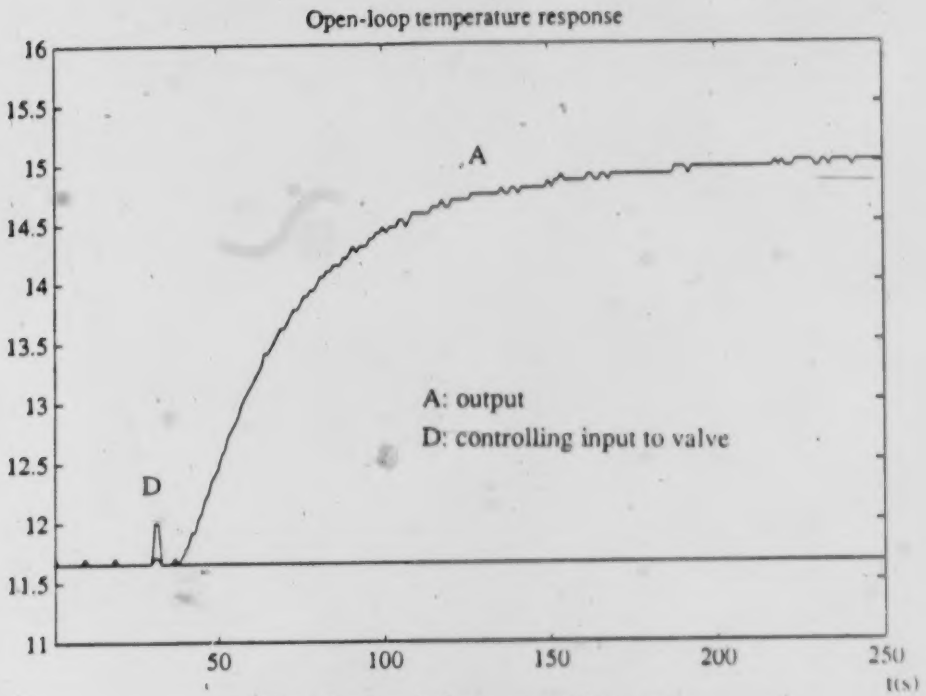


Fig.4.2b: A typical process reaction curve

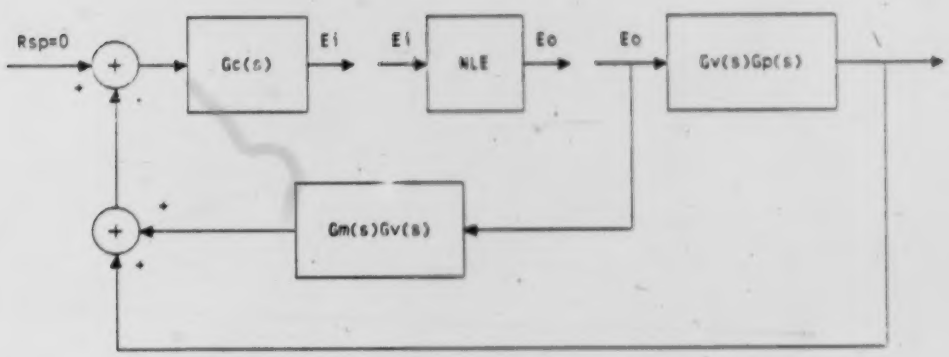


Fig.4.3: Block diagram of a nonlinear feed back system

CONTROLLER PARAMETER (PID)	ZIEGLER & NICHOLS	FINE TUNING	RELAY DEADZONE [V]	TOLERANCE [L/Min]
Kp [V/(L/Min)]	0.108	0.09	0.2	∓ 0.5
Kd [V.s/(L/Min)]	0.327	0.1635		
Ki [V/(L.s/Min)]	0.0089	0.0045		

Table 12: Controller's settings for the flow control loop.

SET POINT Tsp [°C]	PROCESS GAIN K [°C/(V.sec)]	TIME CONSTANT T [sec]	DEAD* TIME Td [sec]
14	0.0110	30	14
16	0.0120		
18	0.0100		
20	0.0035		

Table 13: Process characteristics of the temperature control loop.

FIGURE	ESTIMATED PARAMETER						EFFECTS
	Ke [°C/Vs]	Error [%]	Te [s]	Error [%]	Tde [s]	Error [%]	
4.3.2a&b	0.0084	-30	30	0	14	0	destabilizing, LC
4.3.3a&b	0.0156	+30	30	0	14	0	stabilizing
4.3.4a&b	0.0120	0	39	+30	14	0	destabilizing
4.3.5a&b	0.0120	0	21	-30	14	0	stabilizing
4.3.6a&b	0.0120	0	30	0	11	-21	unaffected
4.3.7a&b	0.0120	0	30	0	17	+21	unaffected
4.3.8a&b	0.0120	0	30	0	8	-40	destabilizing, LC
4.3.9a&b	0.0120	0	30	0	20	+40	destabilizing, LC
4.3.10a&b	0.0108	-10	33	+10	11	-21	destabilizing, LC
4.3.11a&b	0.0132	+10	27	-10	11	-21	stabilizing

Table 14 : Effects of modeling errors on process stability.

CONTROLLER	Kp [V/°C]	Ki [V/(°C.s)]	Kd [V.s/°C]
P	≤ 3	0	0
PD	≤ 3	0	3
PID	≤ 3	≤ 0.02	3

Table 15: Controller's settings for the temperature control loop.

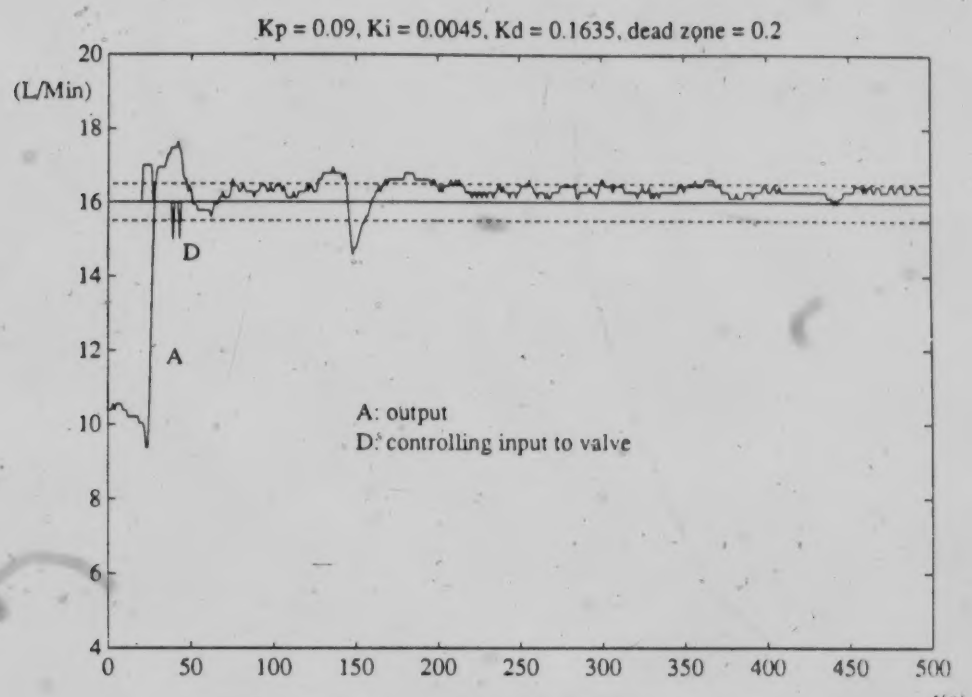


Fig.4.1.1: Flow control (Fsp = 16L/Min)

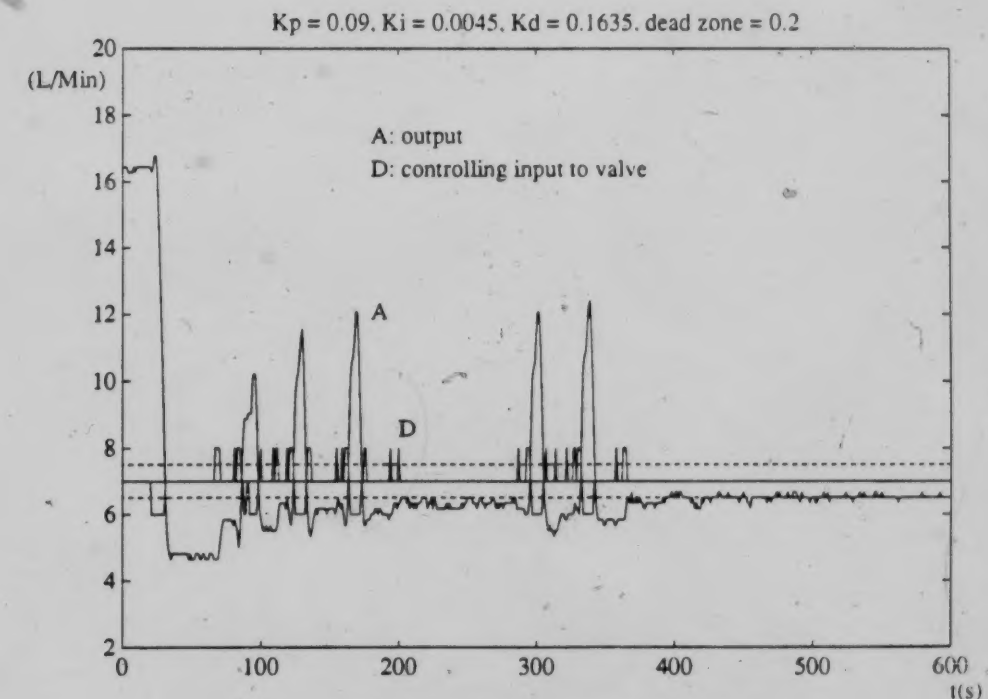


Fig.4.1.2: Flow control (Fsp = 7L/Min)

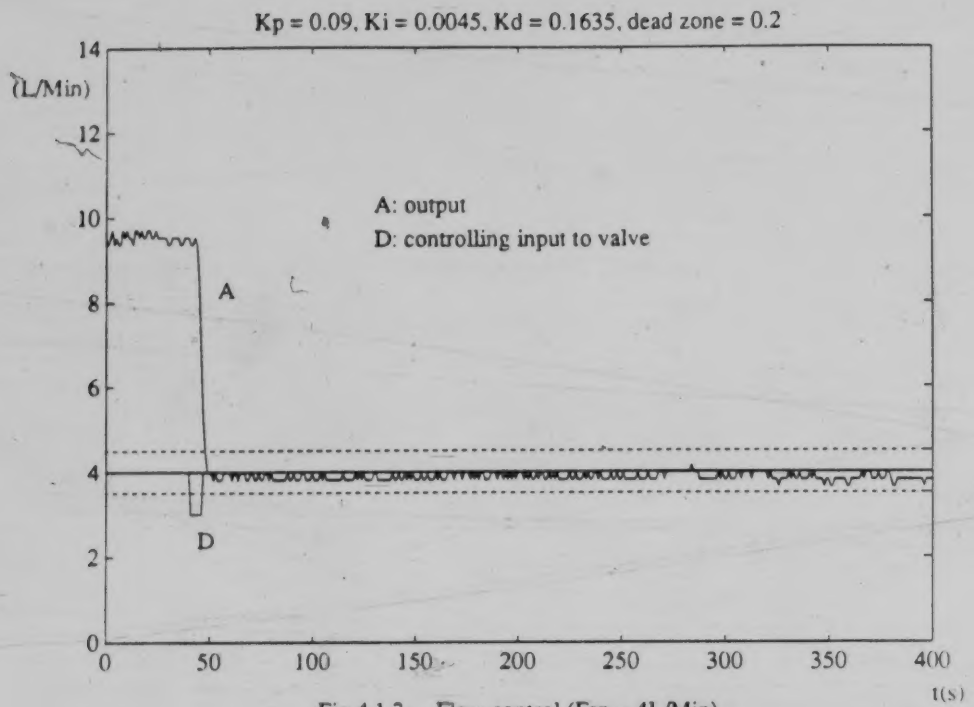


Fig.4.1.3: Flow control (Fsp = 4L/Min)

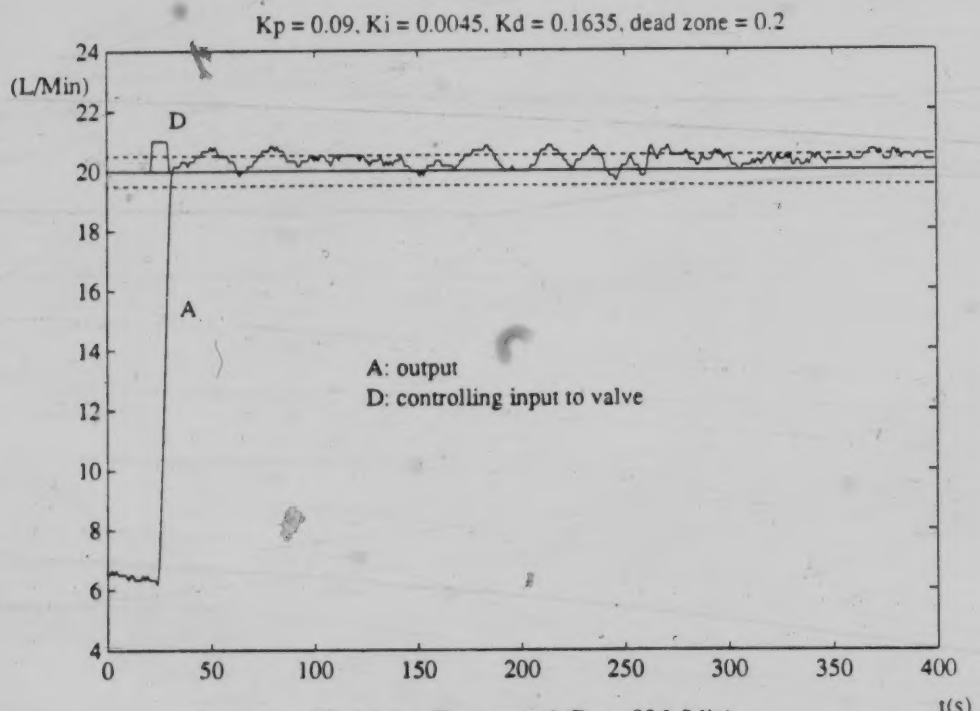


Fig.4.1.4: Flow control (Fsp = 20 L/Min)

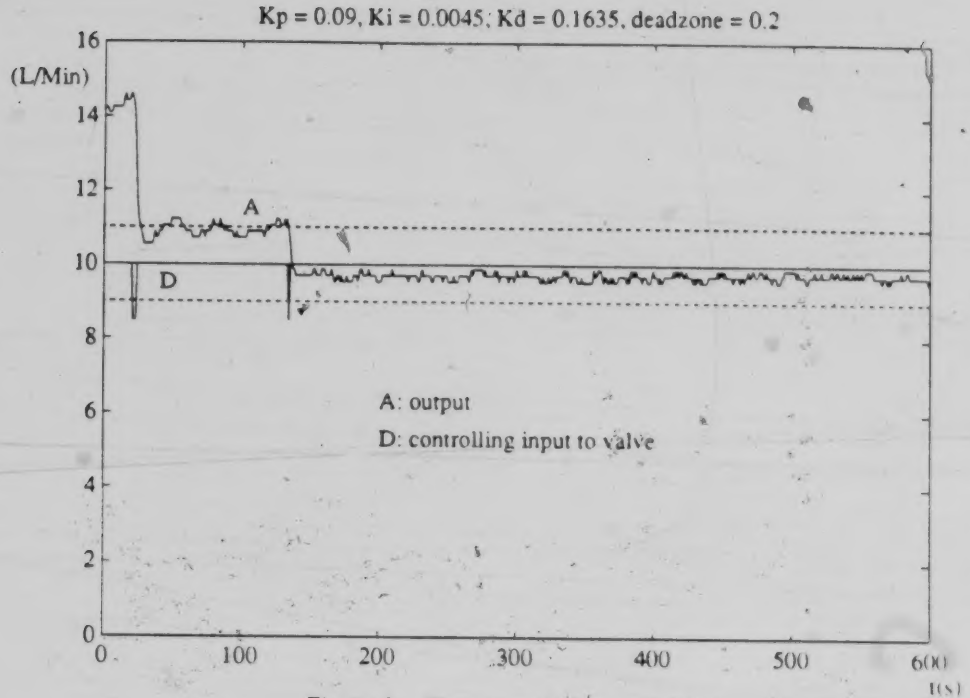


Fig.4.1.5: Flow control (Fsp = 10L/Min)

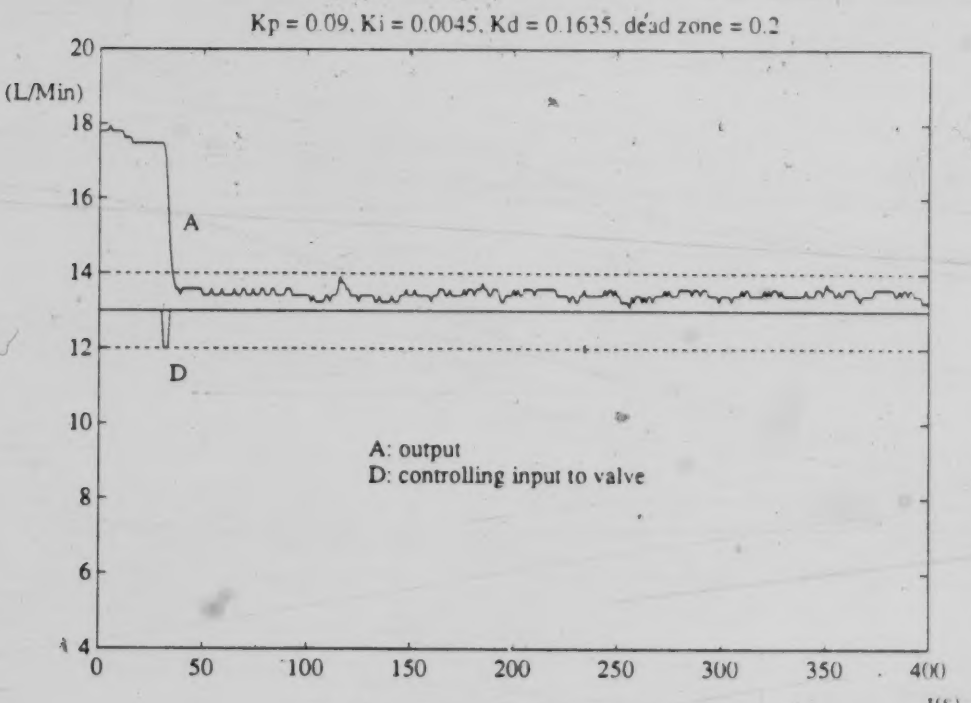


Fig.4.1.6: Flow control (Fsp = 13L/Min)

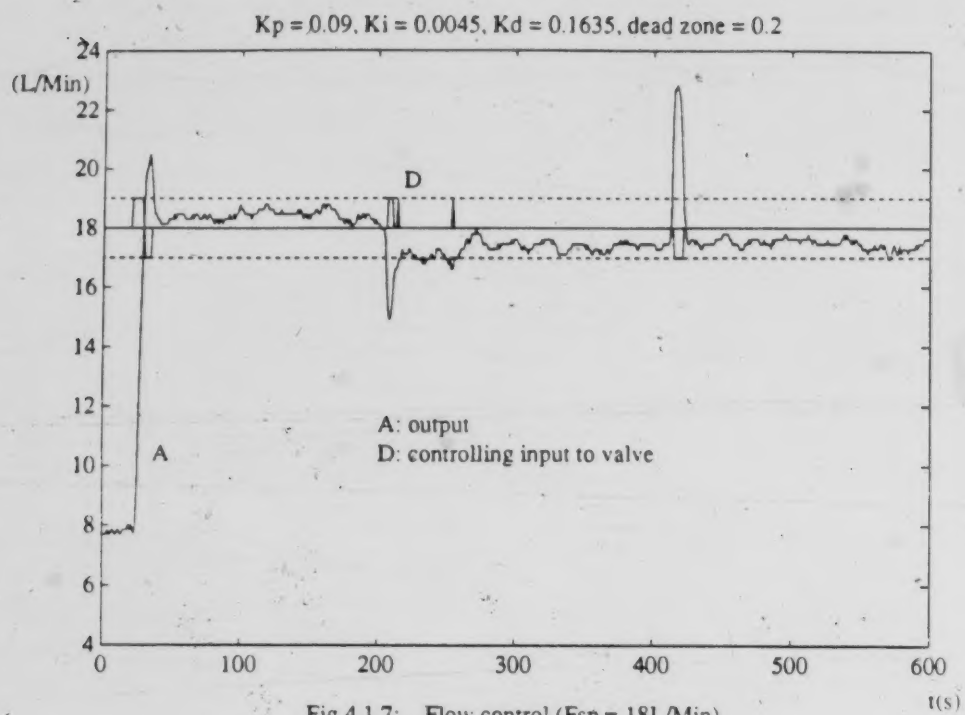


Fig.4.1.7: Flow control (Fsp = 18L/Min)

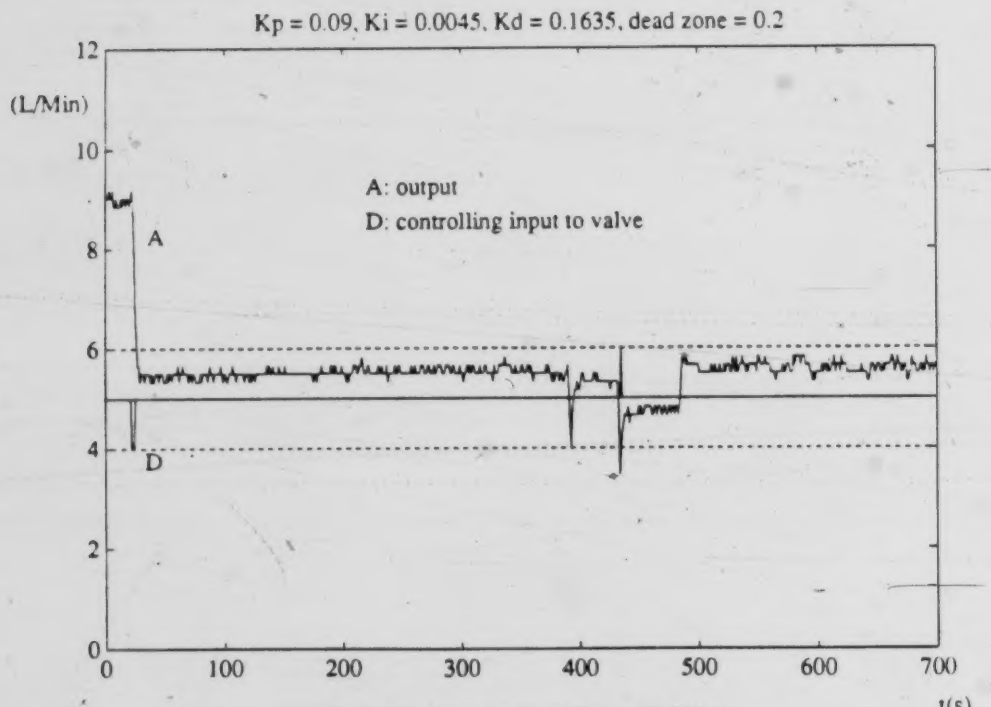


Fig.4.1.8: Flow control (Fsp = 5L/Min)

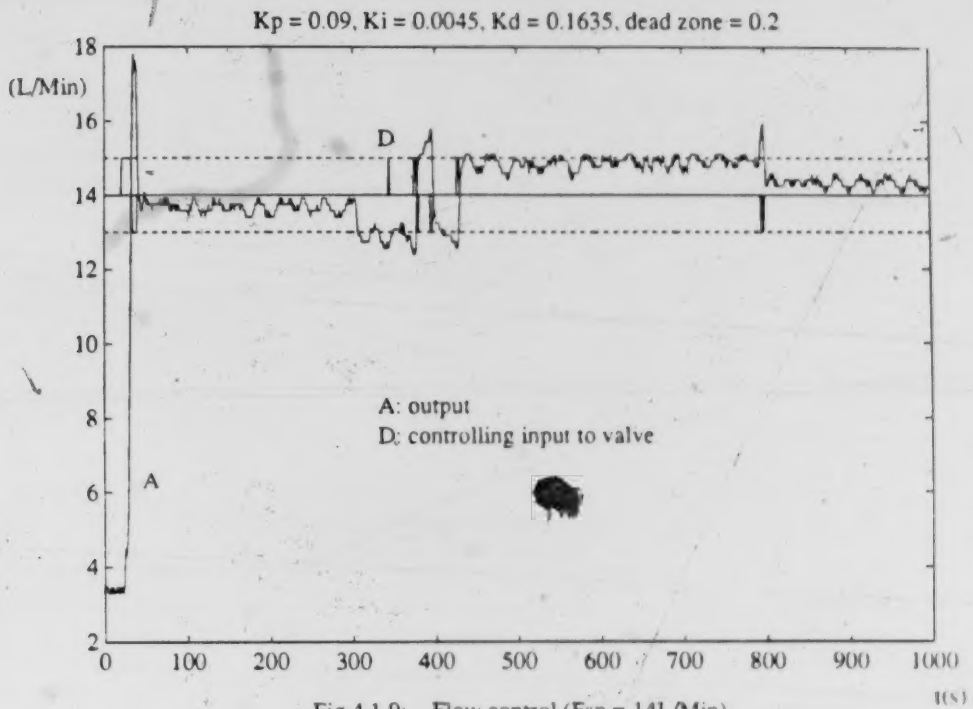


Fig 4.1.9: Flow control (Fsp = 14L/Min)

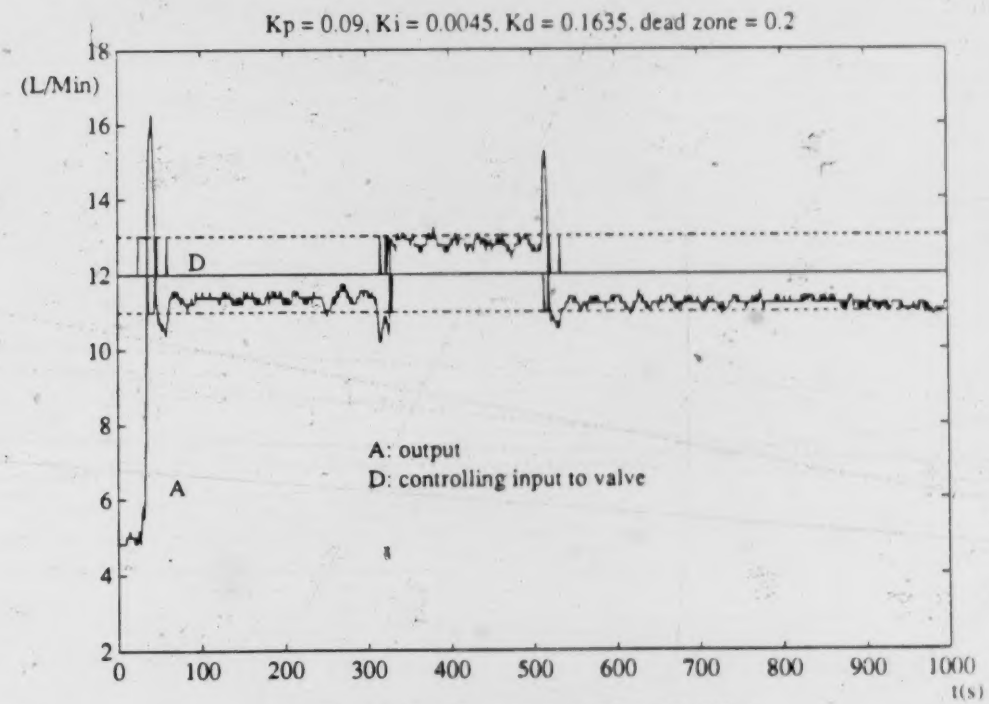


Fig.4.1.10: Flow control (Fsp = 12L/Min)

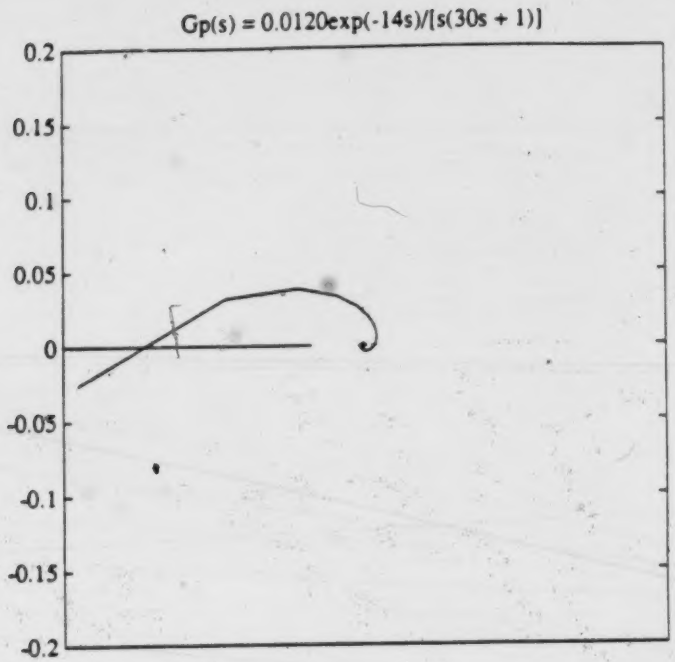


Fig.4.3.1a: Nyquist plot of Gp(s)

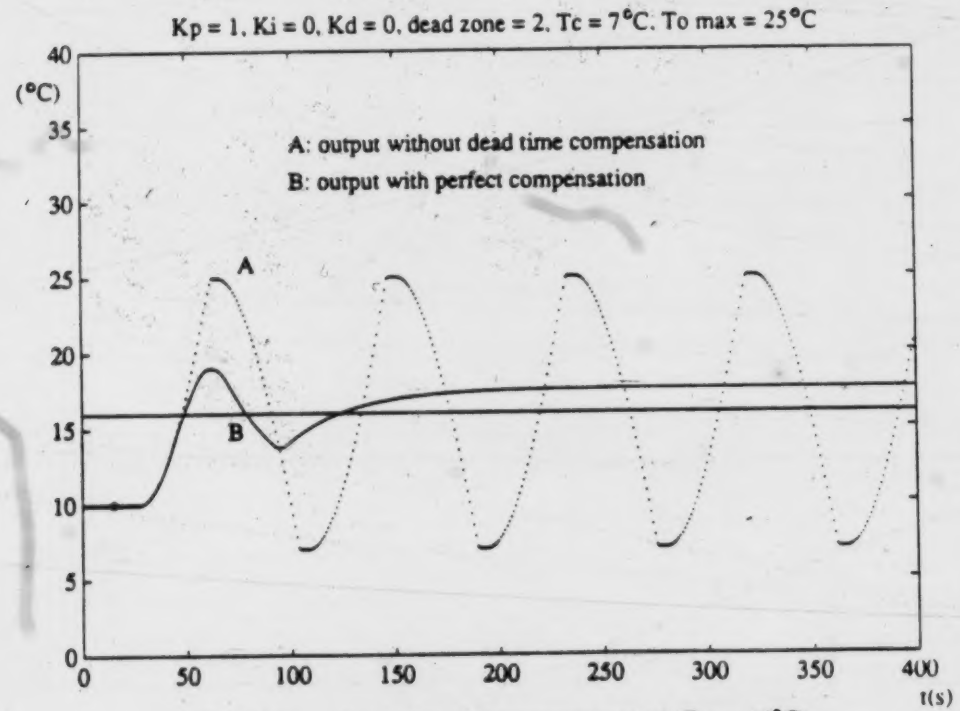


Fig.4.3.1b: Simulation of temperature control (Tsp = 16°C)

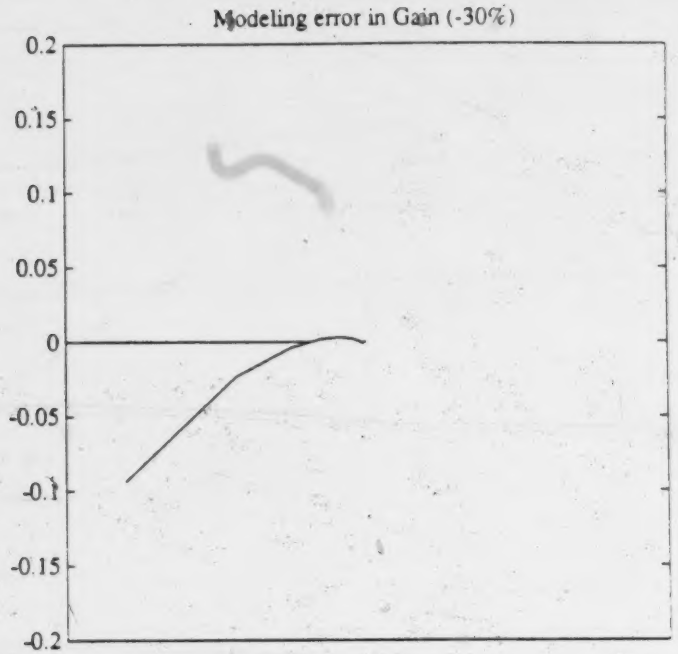


Fig.4.3.2a: Nyquist plot of Gp(s) + Gm(s)

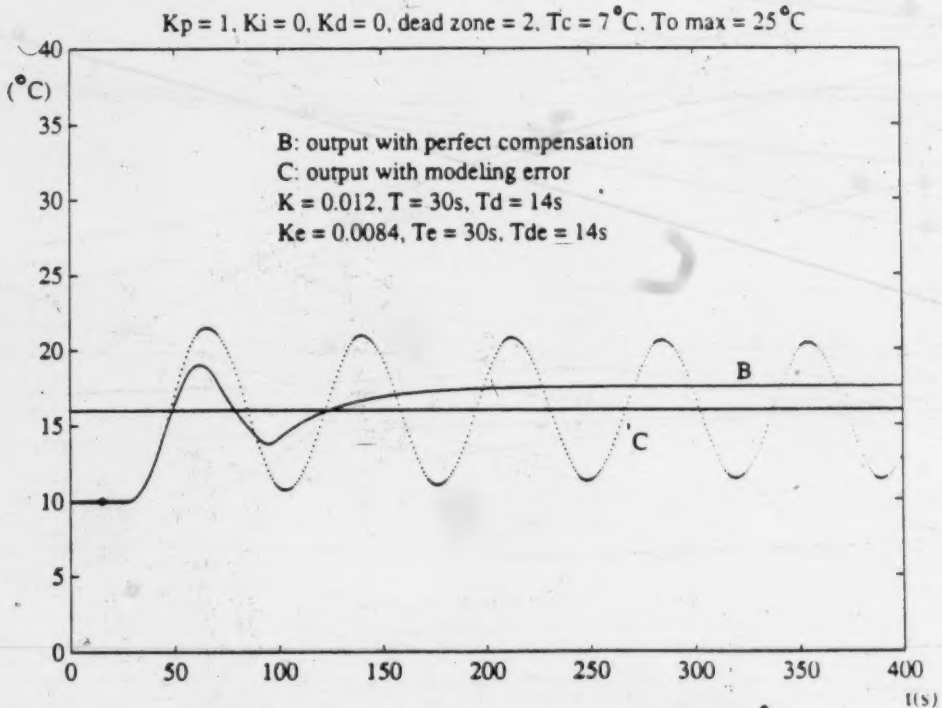


Fig.4.3.2b: Simulation of temperature control (Tsp = 16°C)

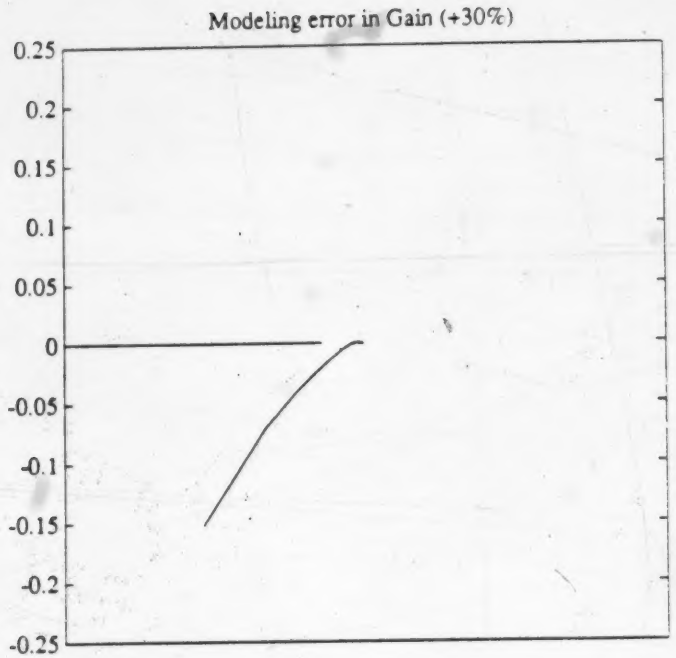


Fig.4.3.3a: Nyquist plot of Gp(s) + Gm(s)

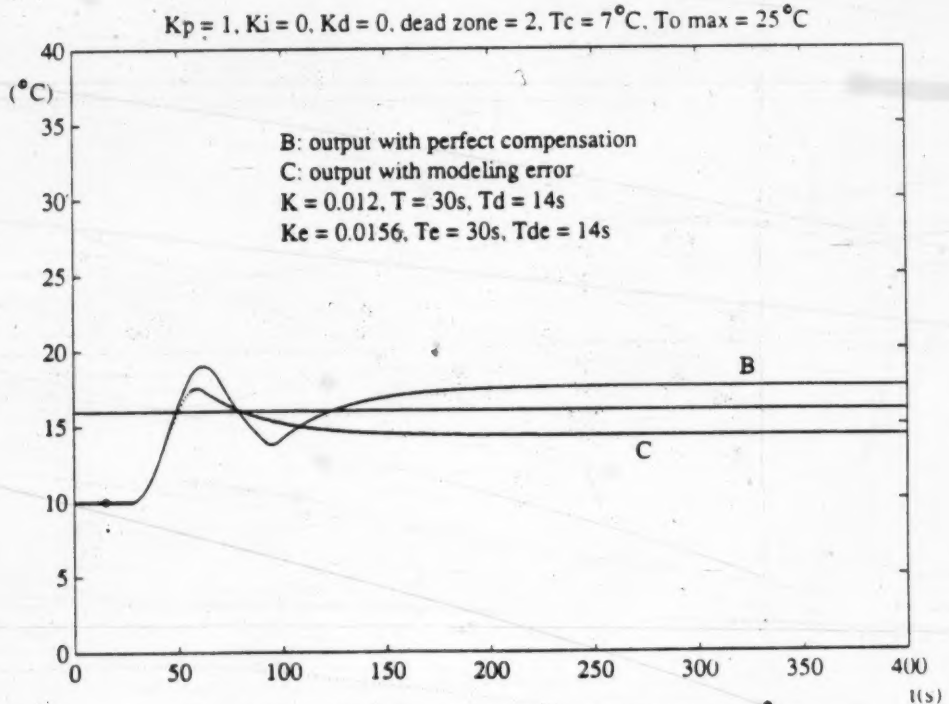


Fig.4.3.3b: Simulation of temperature control (Tsp = 16°C)

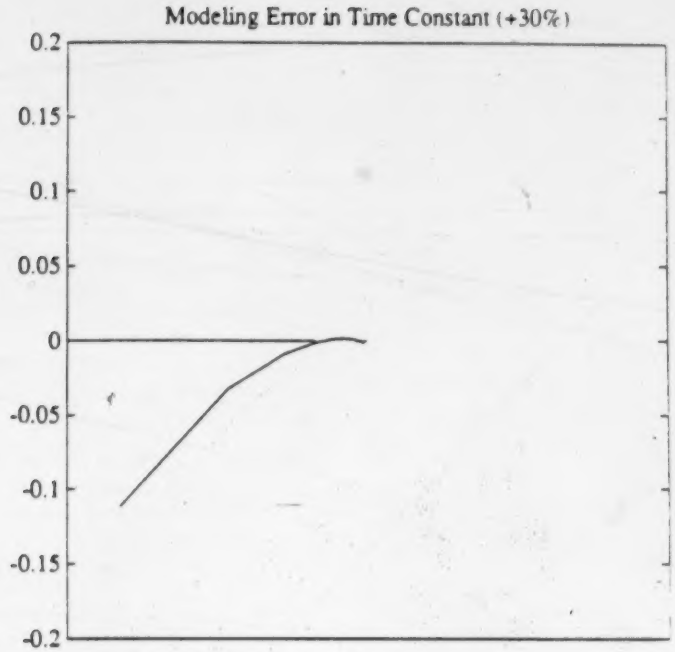


Fig.4.3.4a: Nyquist plot of Gp(s) + Gm(s)

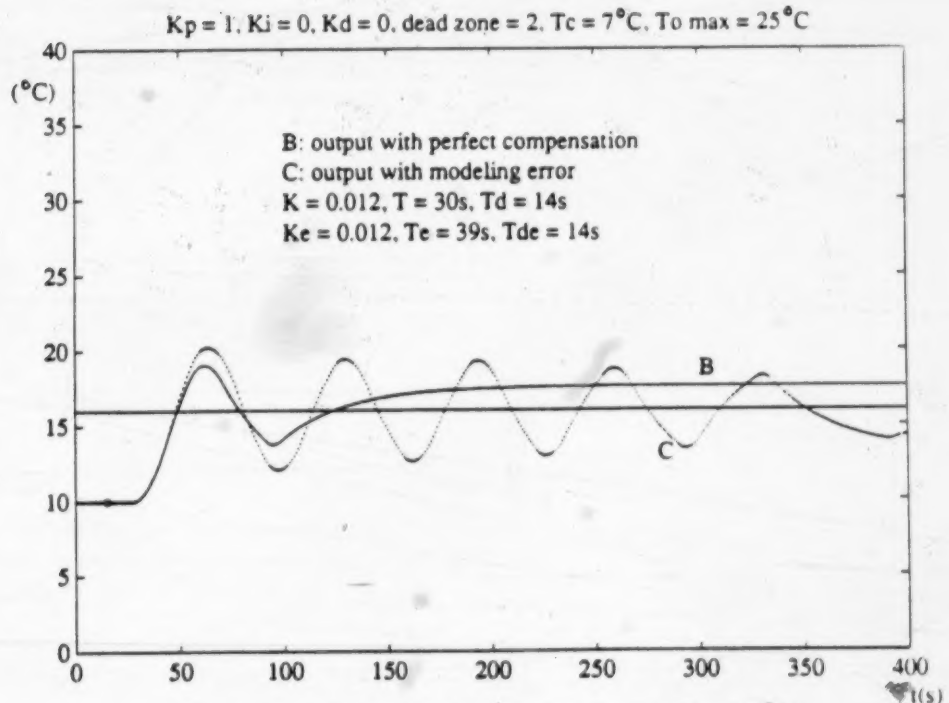


Fig.4.3.4b: Simulation of temperature control (Tsp = 16°C)

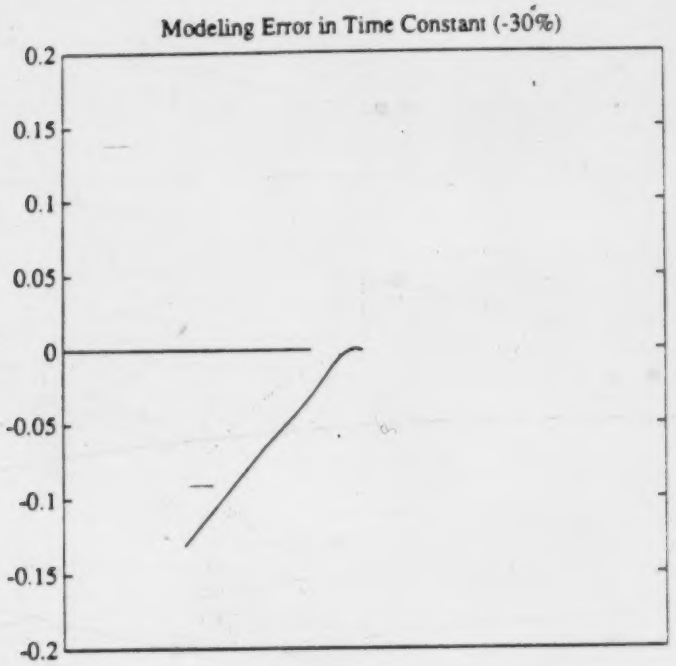


Fig.4.3.5a: Nyquist plot of Gp(s) • Gm(s)

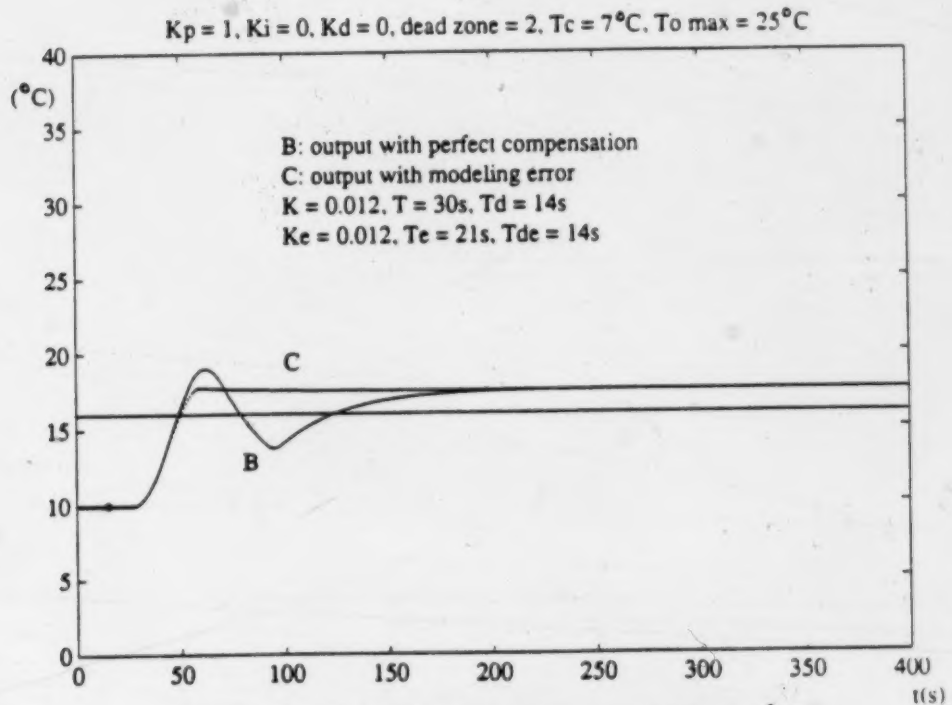


Fig.4.3.5b: Simulation of temperature control (Tsp = 16°C)

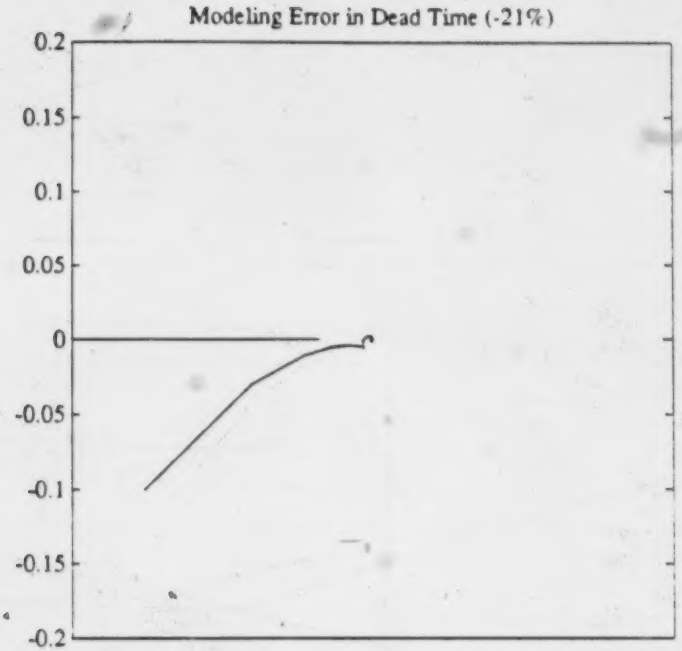


Fig.4.3.6a: Nyquist plot of Gp(s) + Gm(s)

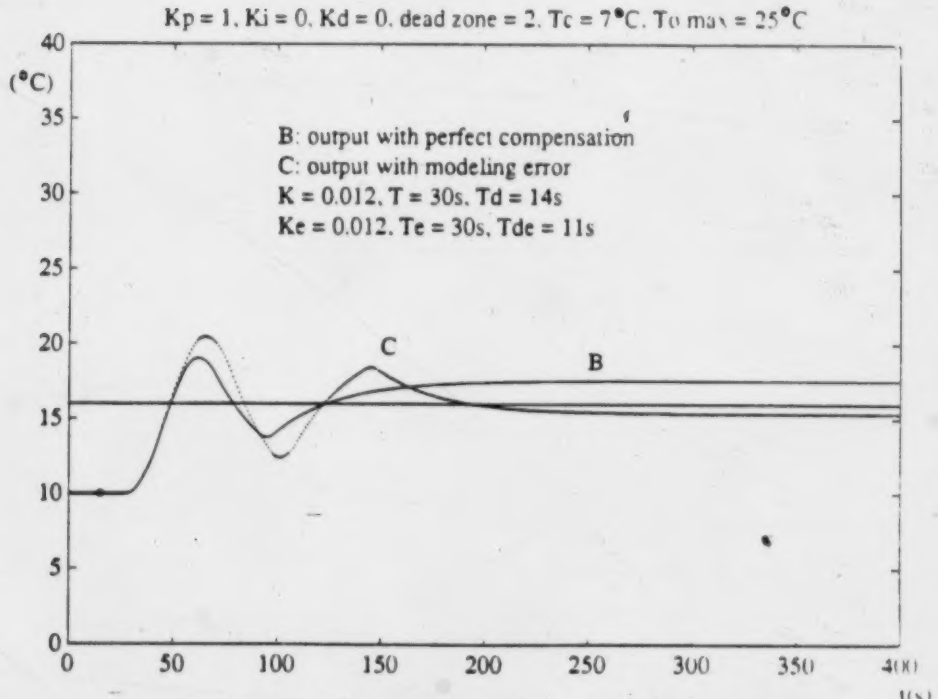


Fig.4.3.6b: Simulation of temperature control (Tsp = 16°C)

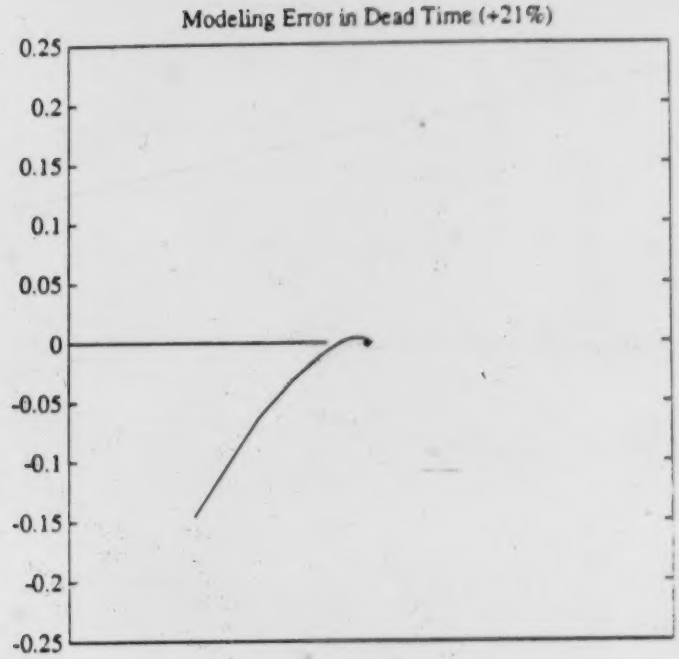


Fig.4.3.7a: Nyquist plot of Gp(s) + Gm(s)

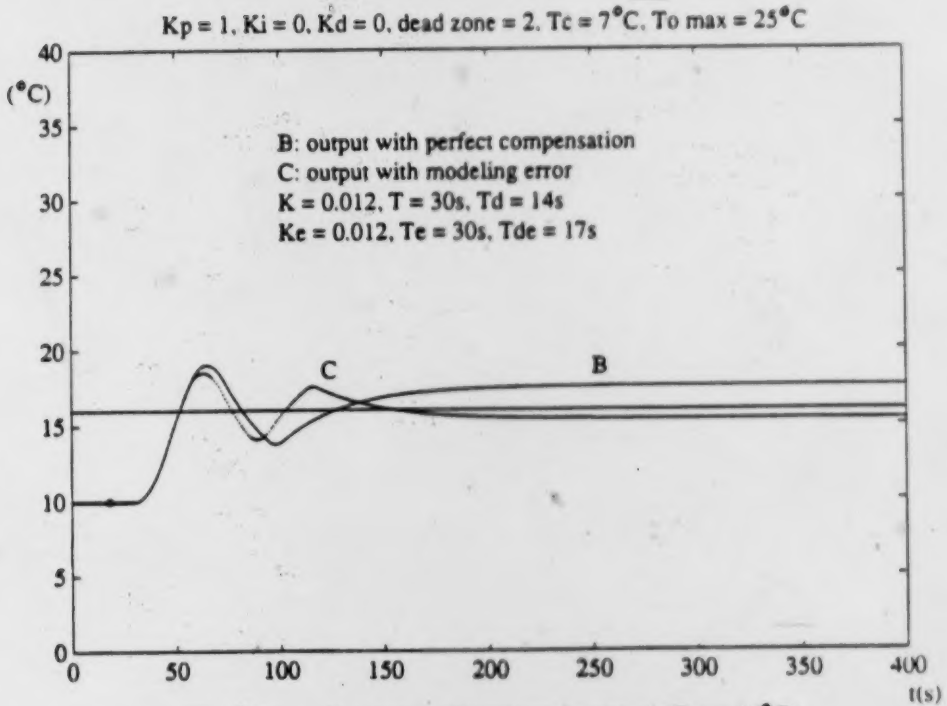


Fig.4.3.7b: Simulation of temperature control (Tsp = 16°C)

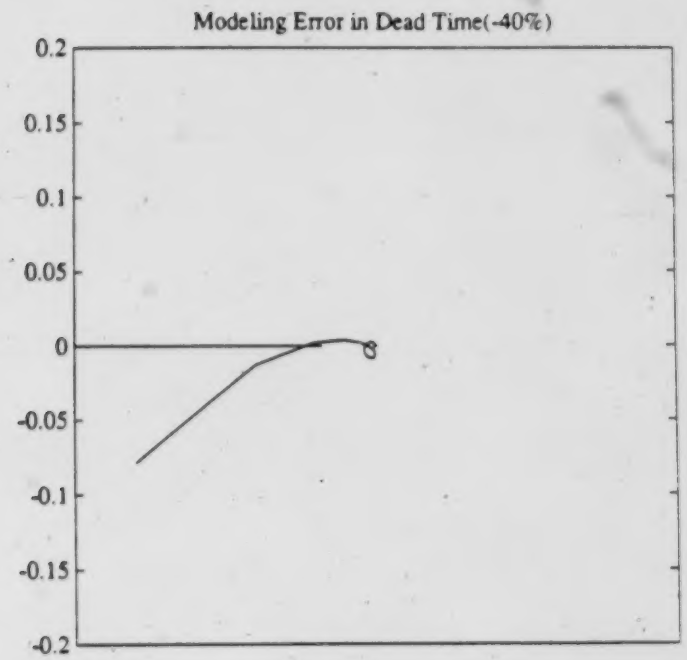


Fig.4.3.8a: Nyquist plot of Gp(s) + Gm(s)

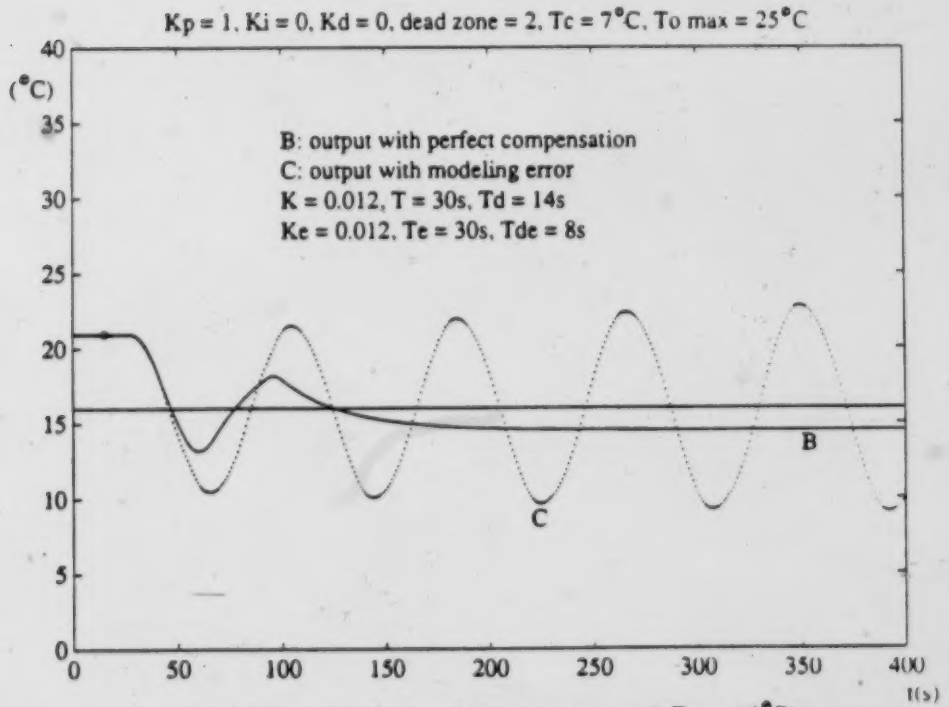


Fig.4.3.8b: Simulation of temperature control (Tsp = 16°C)

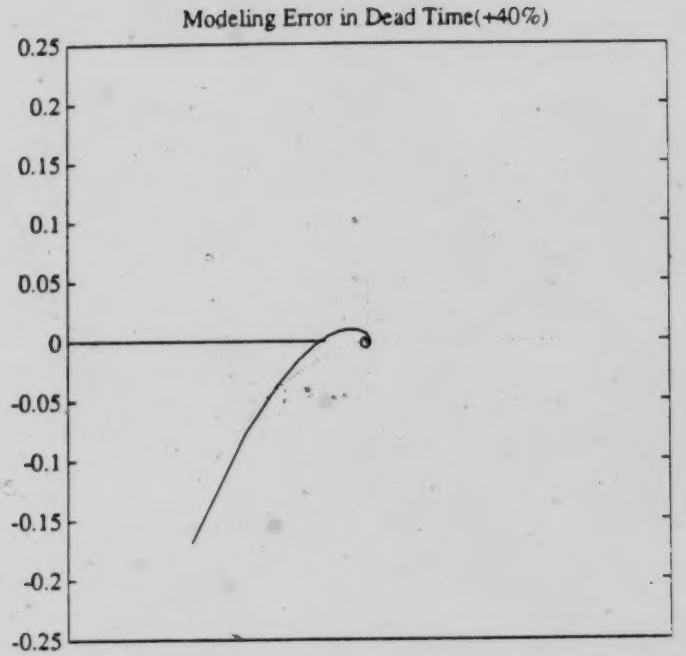


Fig.4.3.9a: Nyquist plot of Gp(s) + Gm(s)

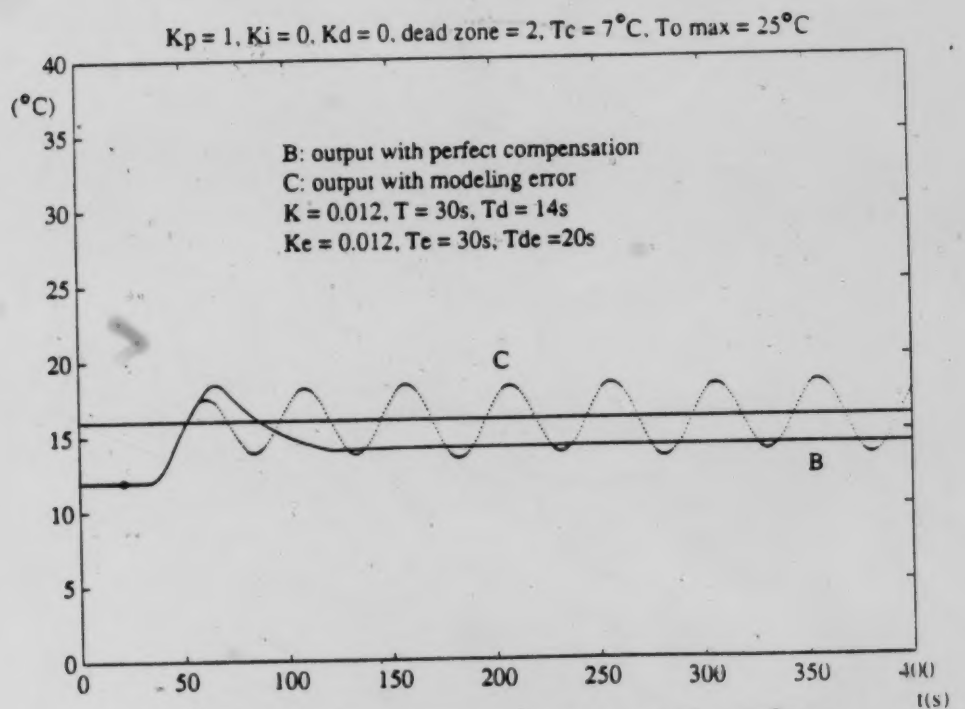


Fig.4.3.9b: Simulation of temperature control (Tsp = 16°C)

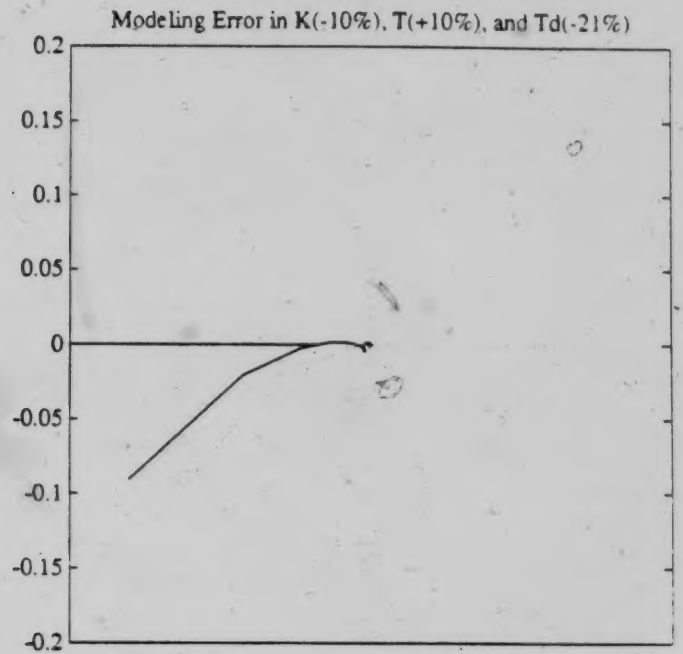


Fig.4.3.10a: Nyquist plot of Gp(s) + Gm(s)

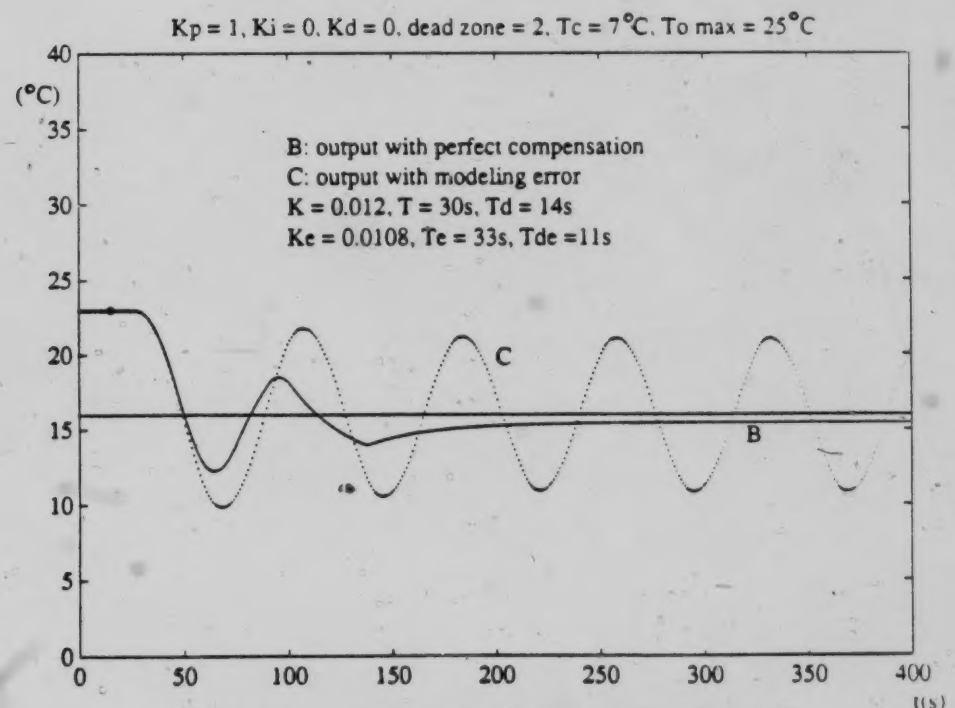


Fig.4.3.10b: Simulation of temperature control (Tsp = 16°C)

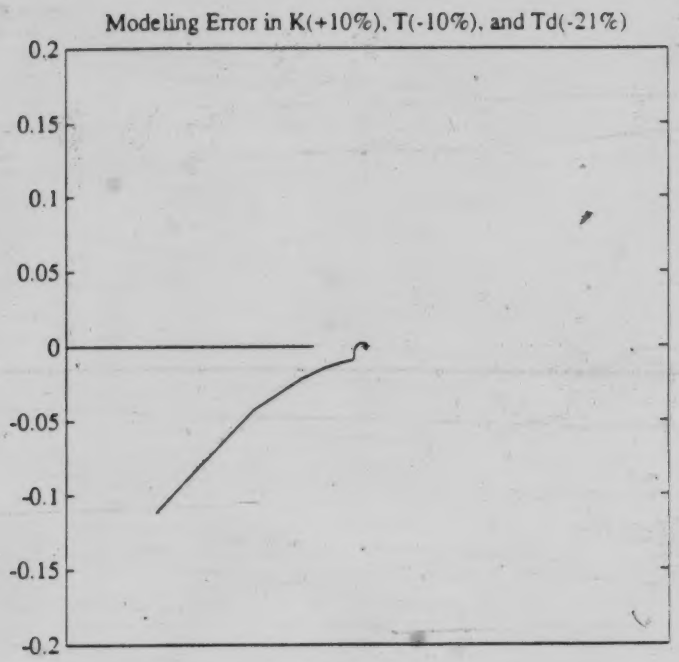


Fig.4.3.11a: Nyquist plot of Gp(s) + Gm(s)

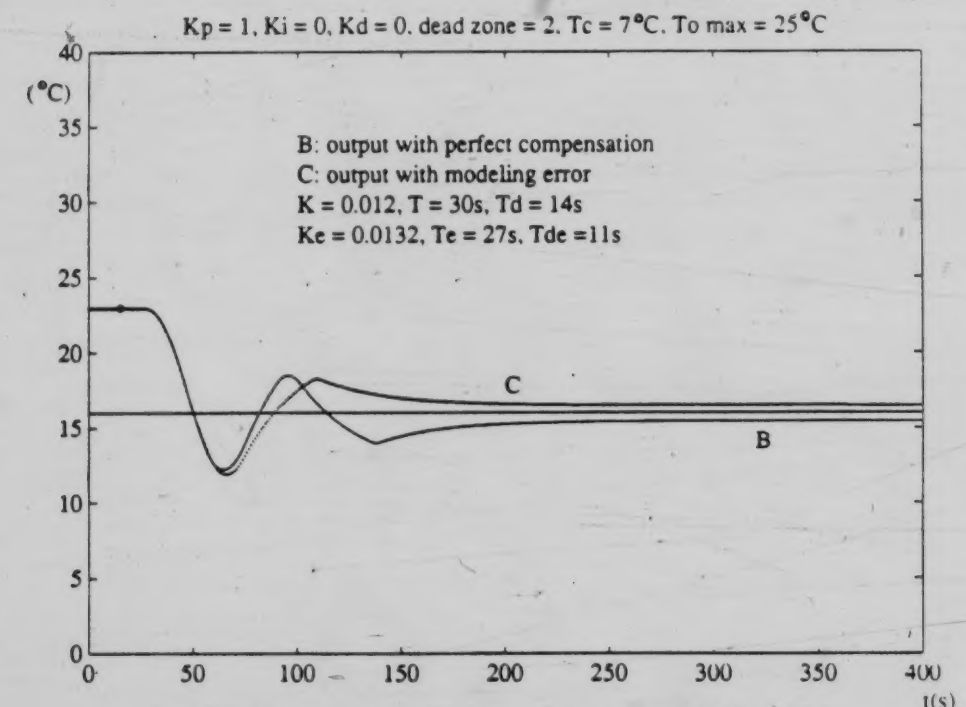


Fig.4.3.11b: Simulation of temperature control (Tsp = 16°C)

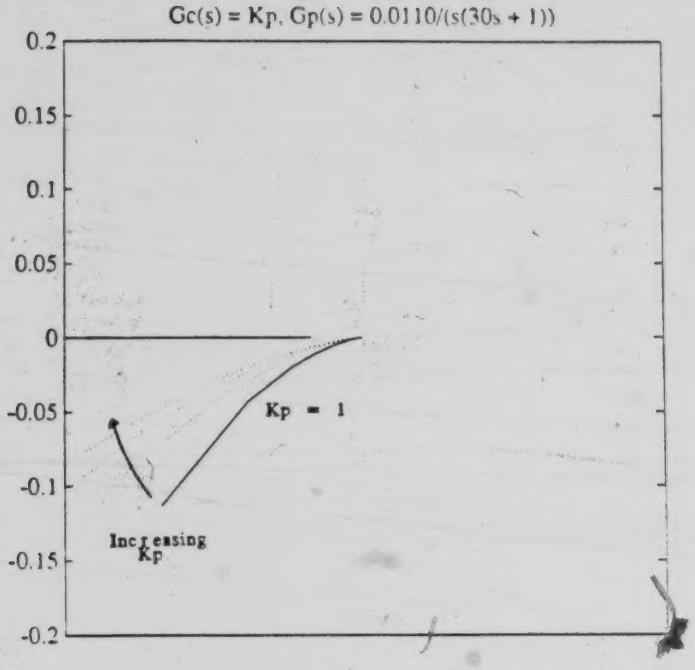


Fig.4.4.1: Nyquist plot of Gc(s)*Gp(s)

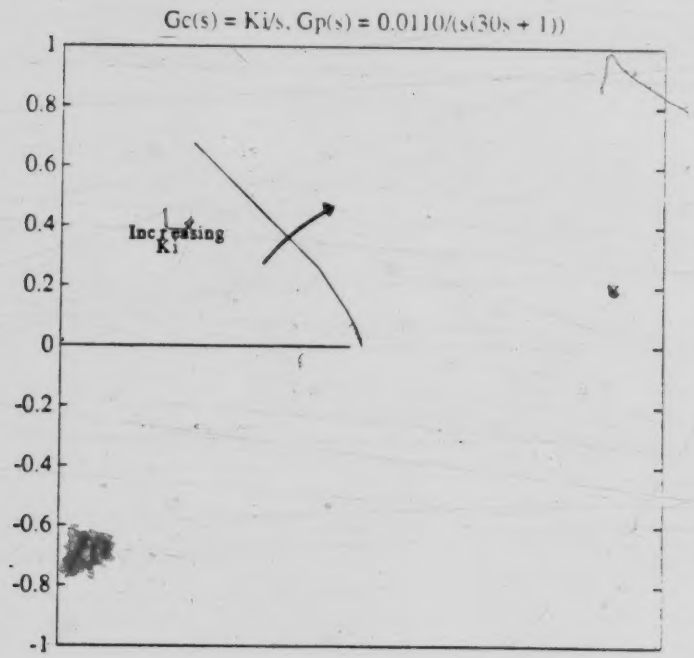


Fig.4.4.2: Nyquist plot of Gc(s)*Gp(s)

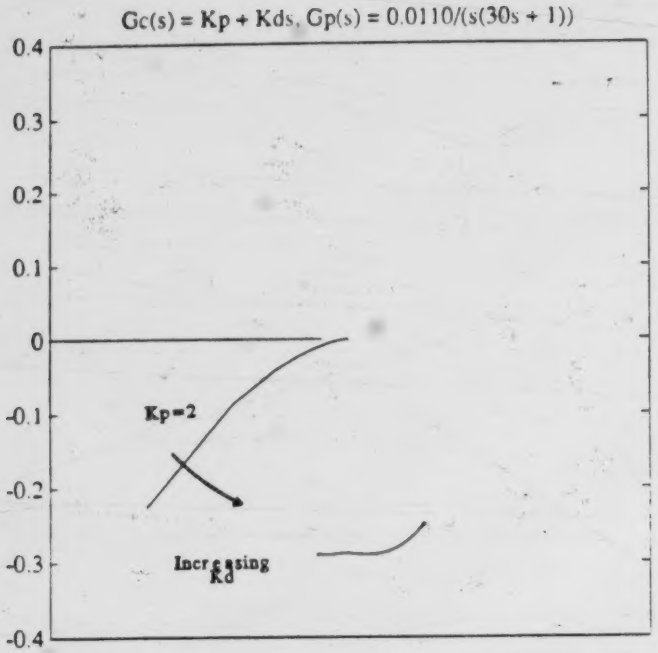


Fig.4.4.3: Nyquist plot of Gc(s)*Gp(s)

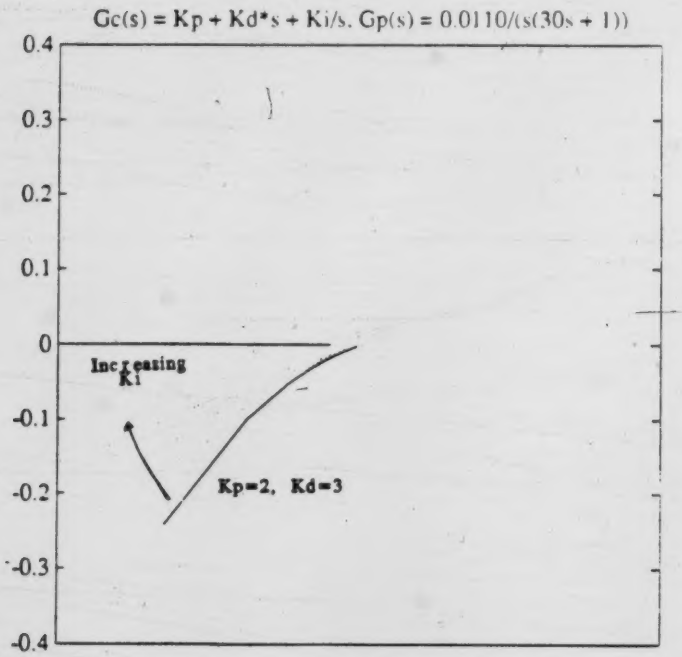


Fig.4.4.4: Nyquist plot of Gc(s)*Gp(s)

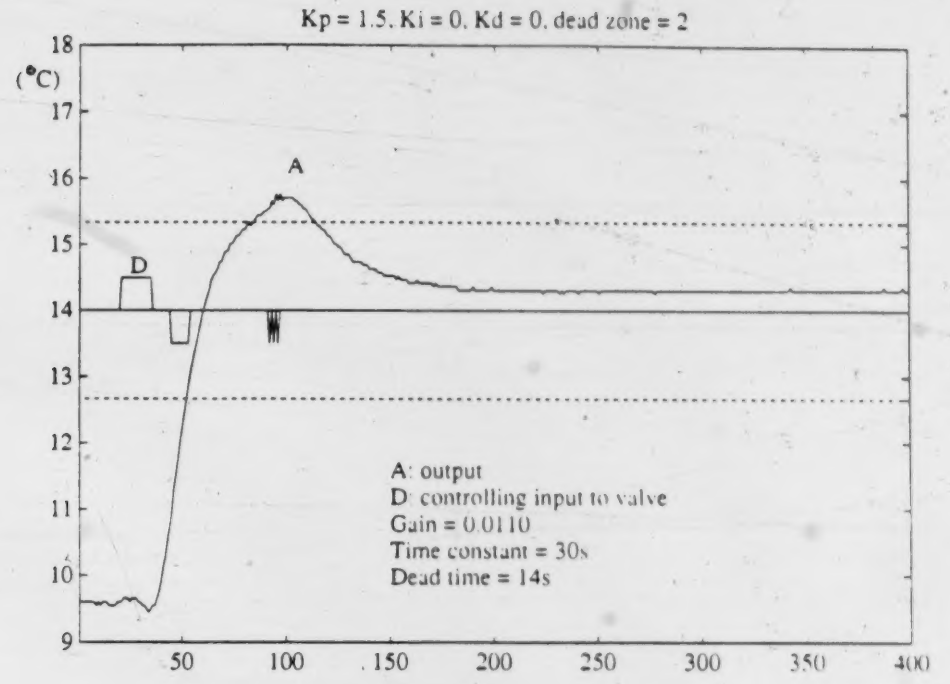


Fig.4.5.1: Temperature control (Tsp = 14 °C)

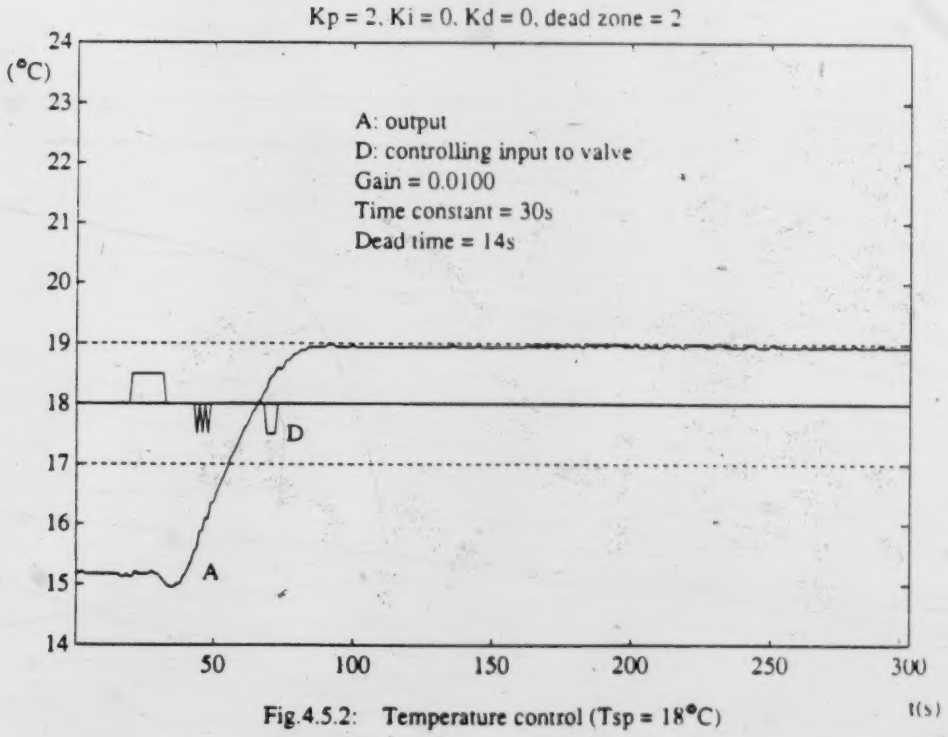


Fig.4.5.2: Temperature control (Tsp = 18°C)

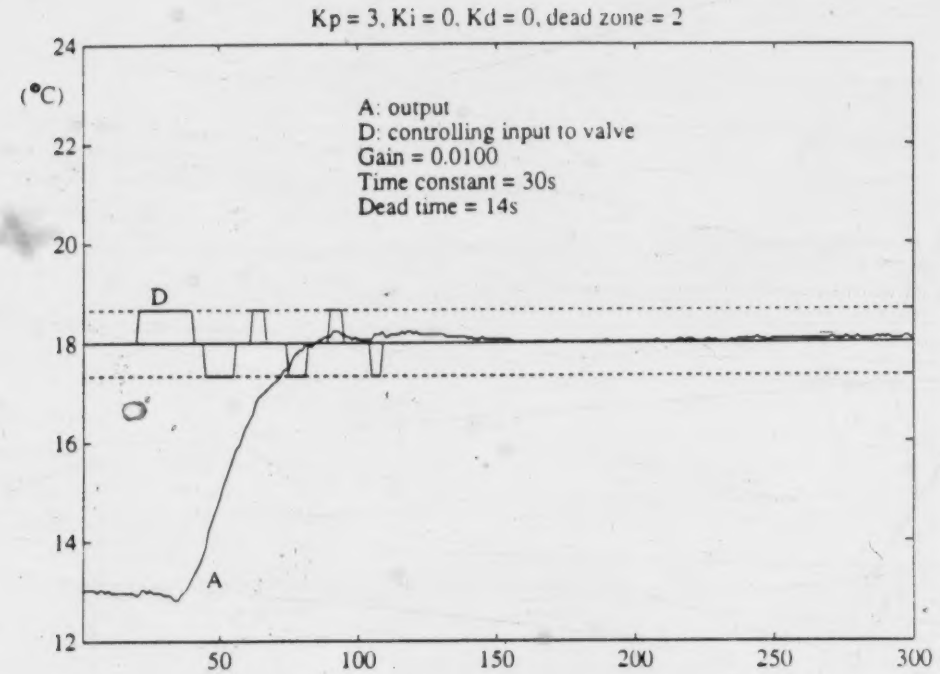


Fig.4.5.3: Temperature control (Tsp = 18°C)

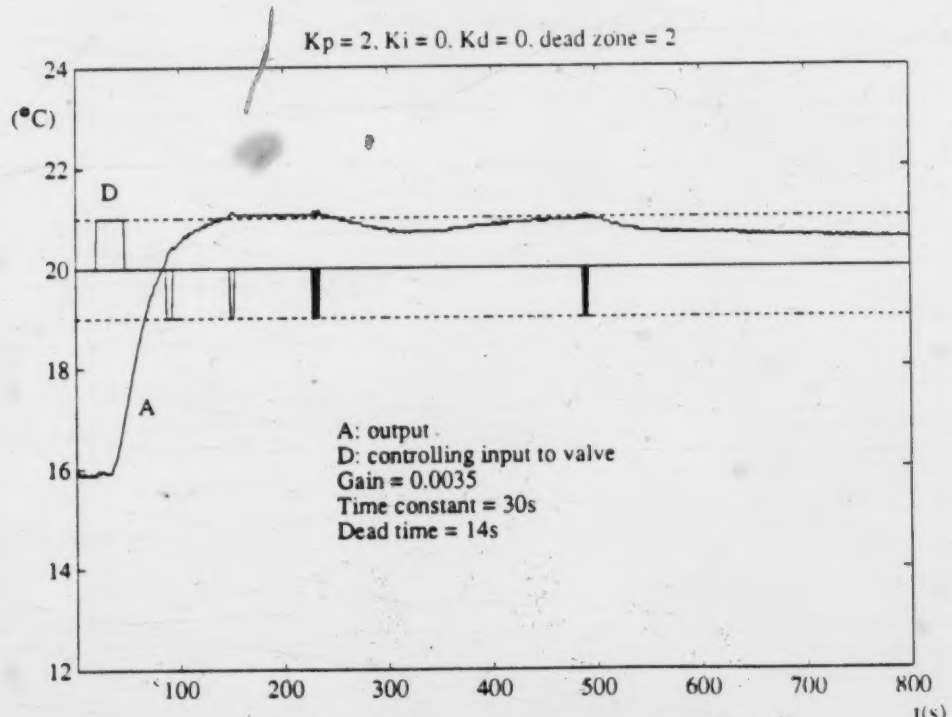


Fig.4.5.4: Temperature control (Tsp = 20°C)

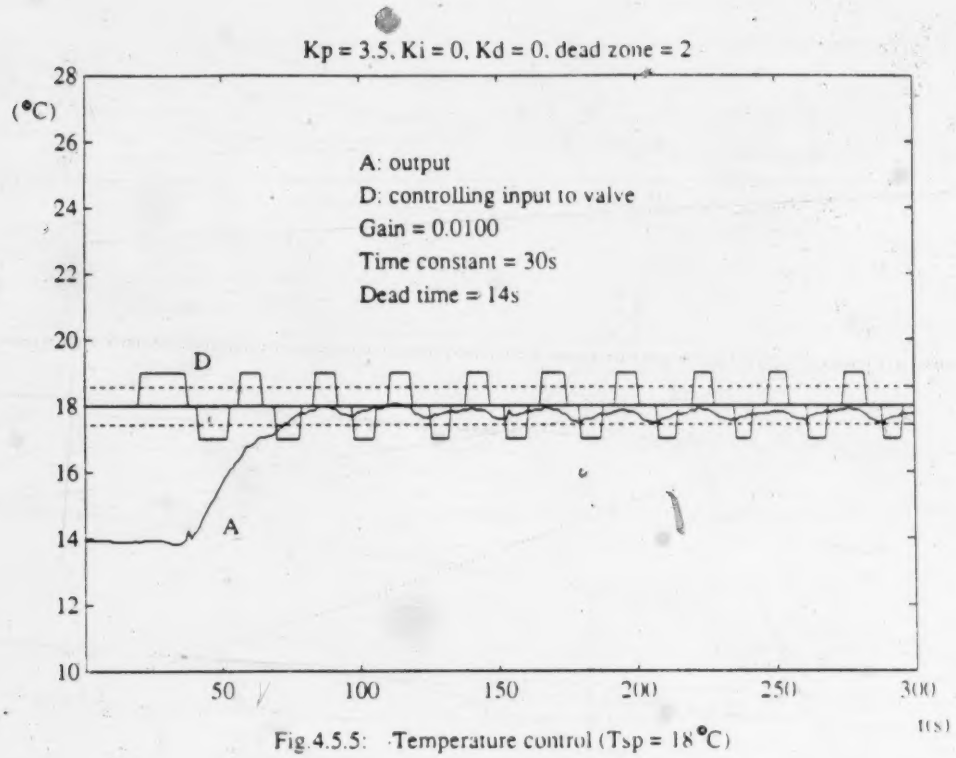


Fig.4.5.5: Temperature control (Tsp = 18°C)

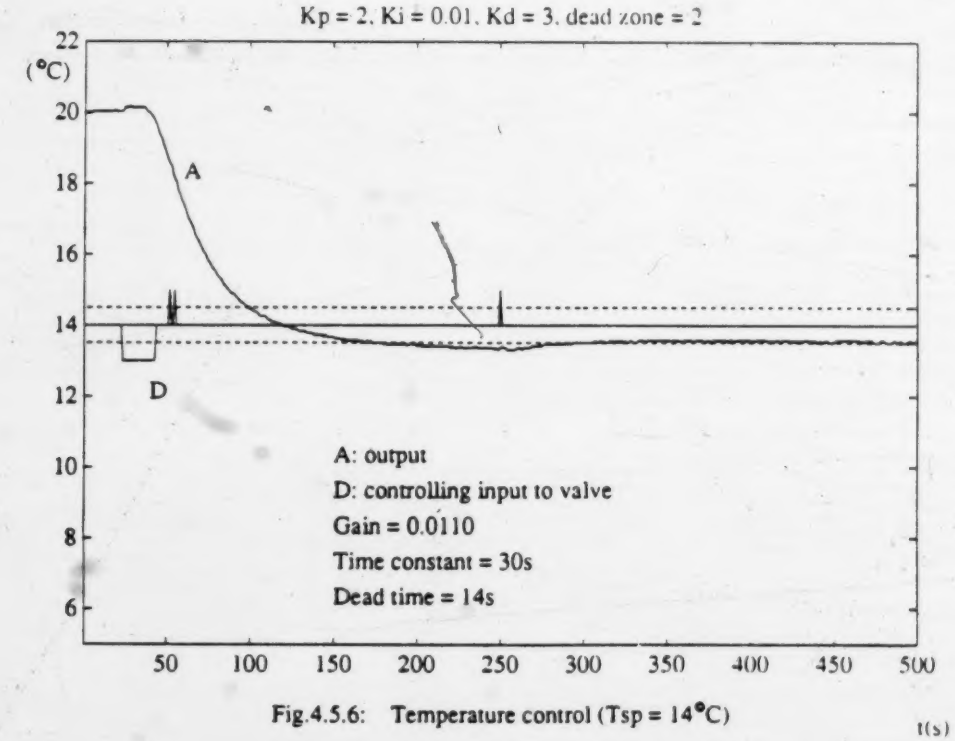


Fig.4.5.6: Temperature control (Tsp = 14°C)

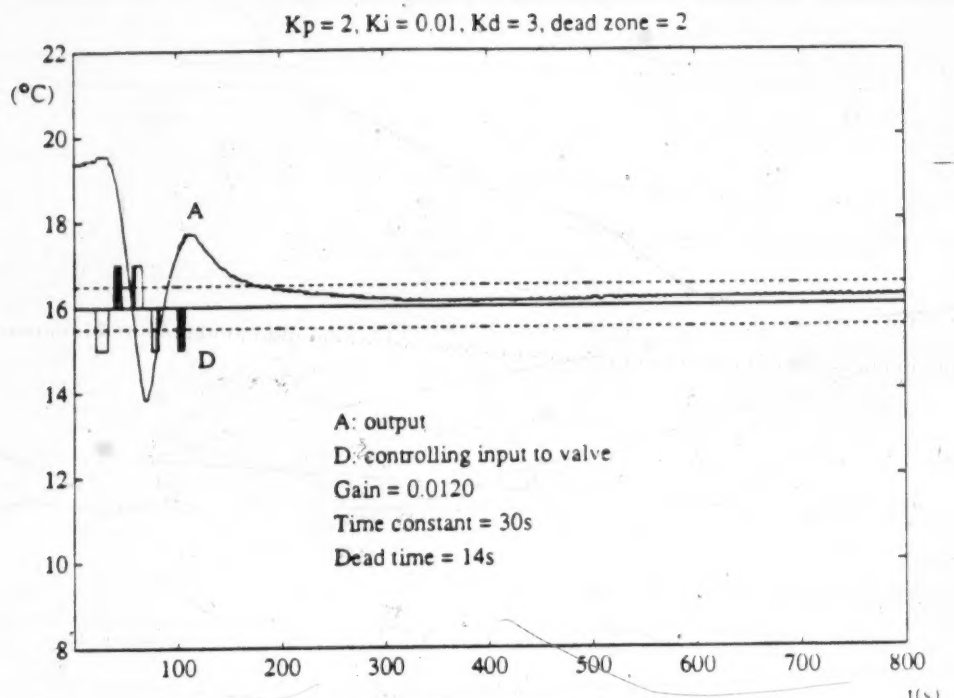


Fig 4.5.7: Temperature control (Tsp = 16°C)

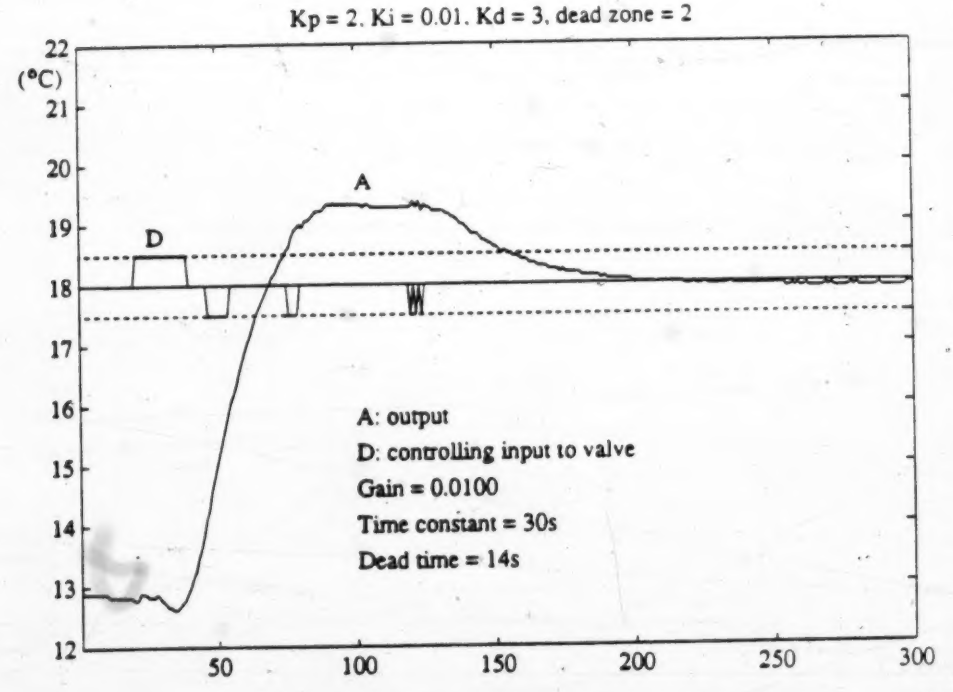


Fig.4.5.8: Temperature control (Tsp = 18°C)

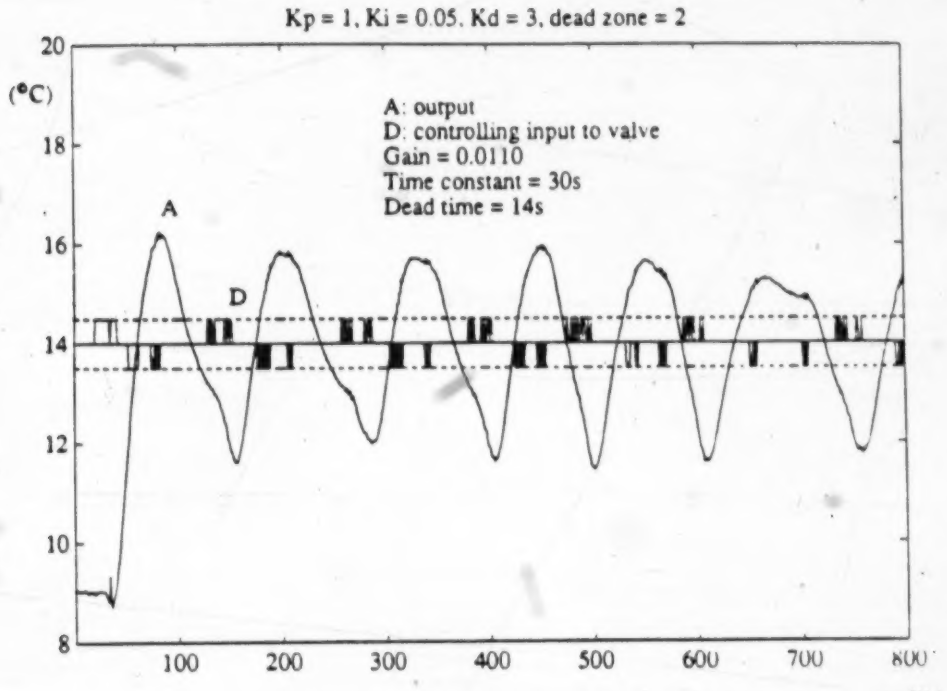


Fig.4.5.9: Temperature control (Tsp = 14°C)

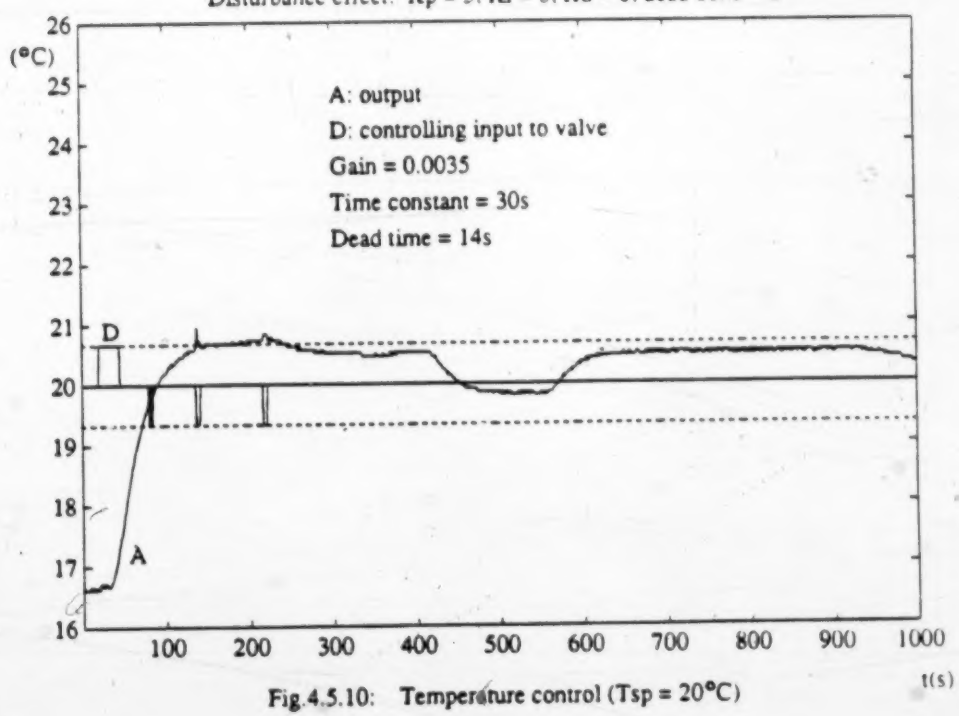


Fig.4.5.10: Temperature control (Tsp = 20°C)

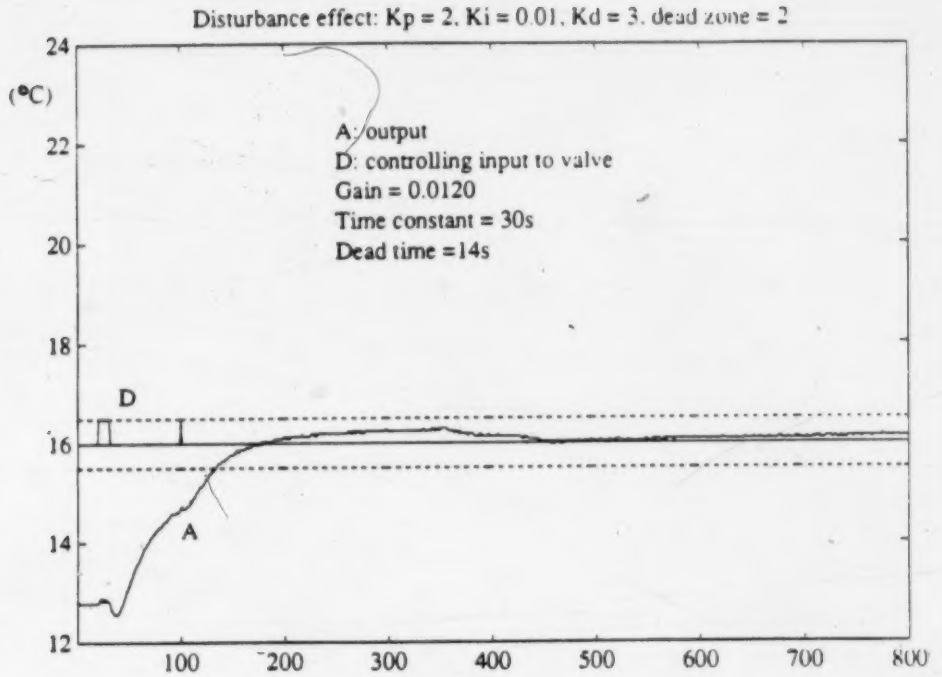


Fig.4.5.11: Temperature control (Tsp = 16°C)

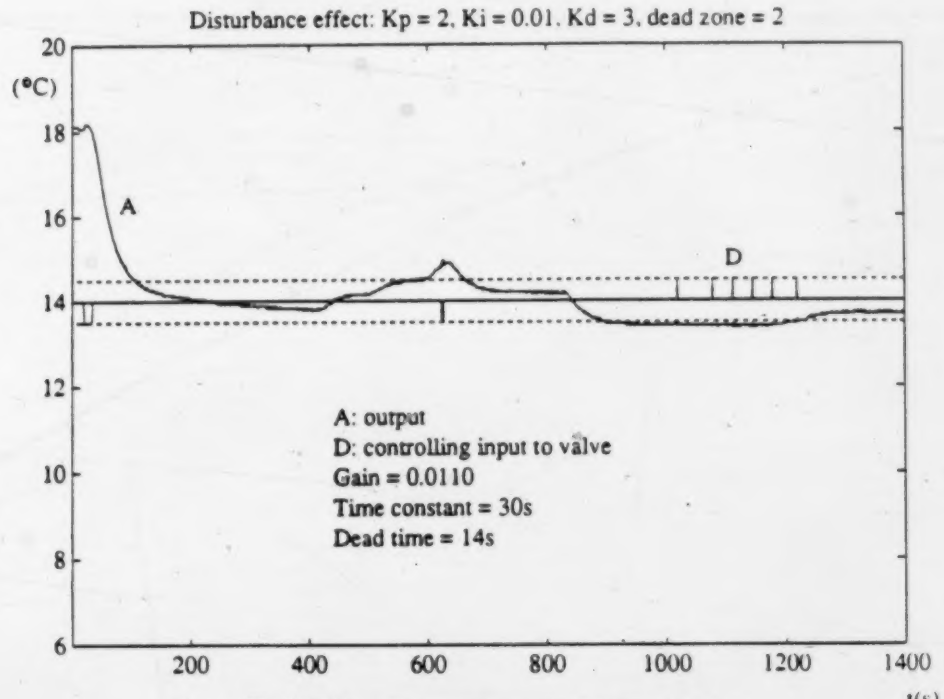


Fig.4.5.12: Temperature control (Tsp = 14°C)

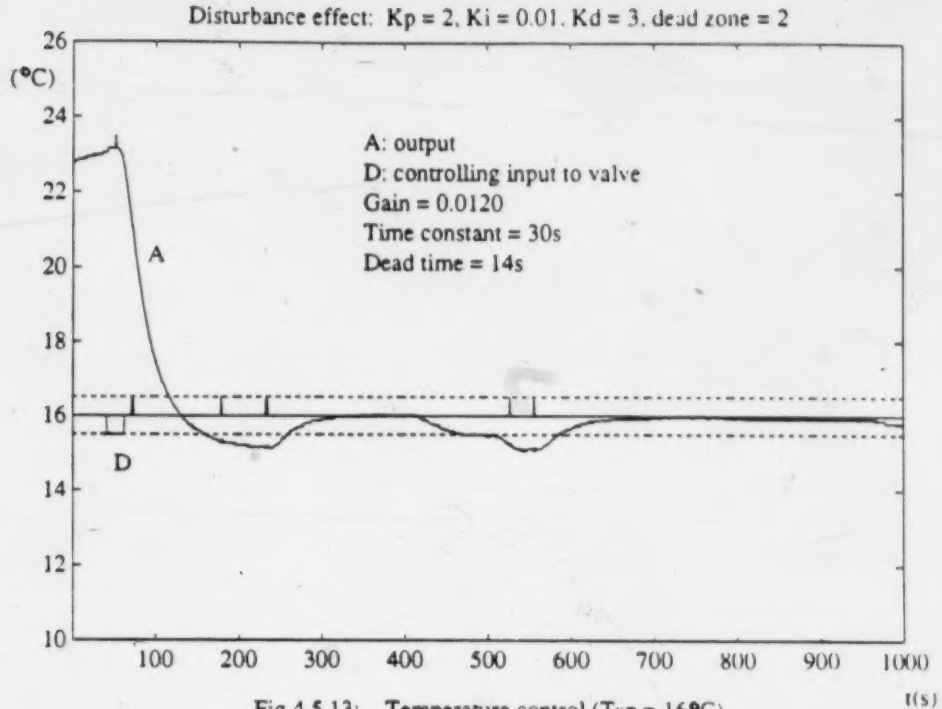


Fig.4.5.13: Temperature control (Tsp = 16°C)

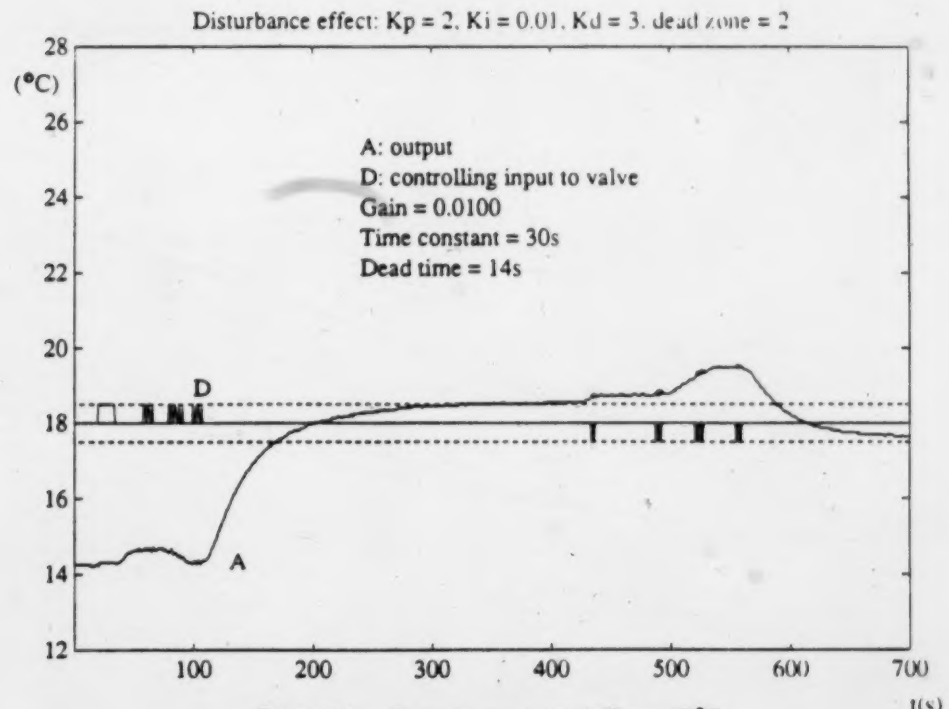


Fig.4.5.14: Temperature control (Tsp = 18°C)

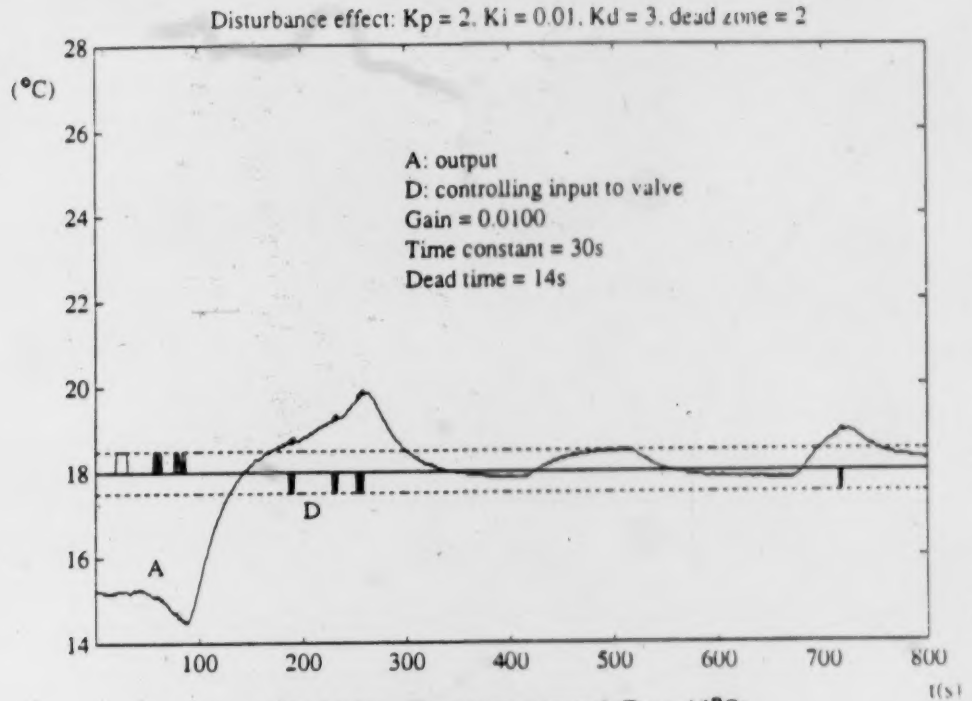


Fig.4.5.15: Temperature control (Tsp = 18°C)

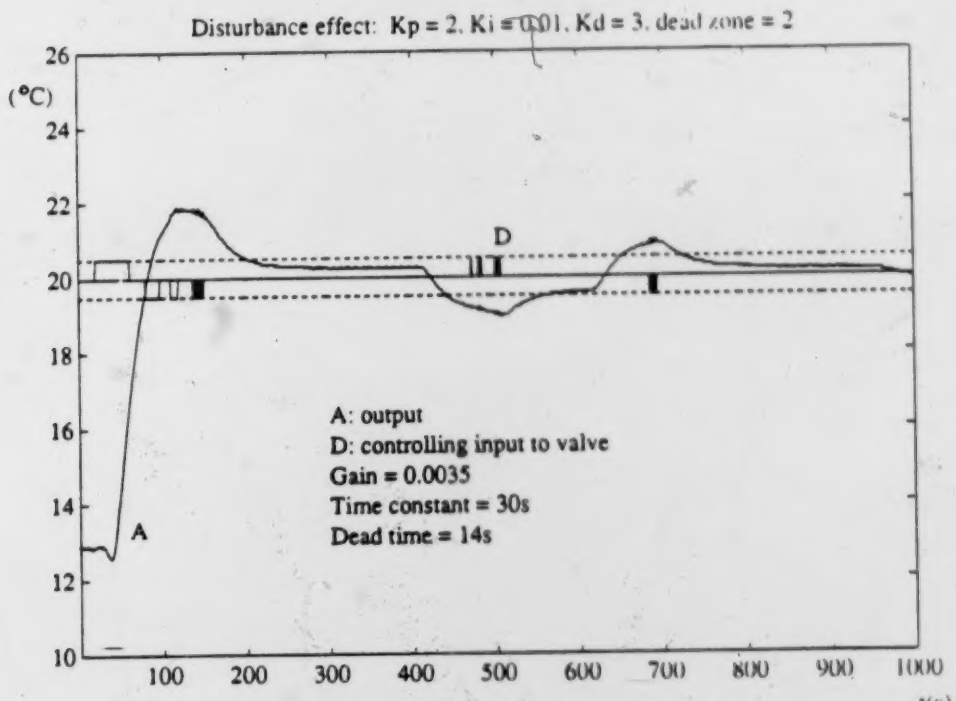


Fig.4.5.16: Temperature control (Tsp = 20°C)

CHAPTER 5
CONCLUSION

The current availability of high performance, low cost personal computers accounts for a rapid growth of their applications in automatic process control. In this research, the IBM PC/XT-based instrumentation and control system facilitates the study and improvement of the heat pump performance through future implementation of optimum control strategies.

For the flow and temperature control loop, the fast sampling period T_{s1} should be set at 1 sec to accommodate the slow motor speed of the valve. On the other hand, due to a significantly large time constant in the heat pump, the sampling period T_{s2} can be chosen to be a multiple of T_{s1}. Such a choice of T_{s2} offers a dual advantage where not only real time overhead but also programming complexity is greatly reduced.

Tuning the flow controller is a simple task because there is no dead time present in the loop, and the Ultimate Sensitivity method is sufficient in this case. Further fine tuning is required in order to obtain the controller's settings which produce the desired performance. Without dead time compensation, the temperature loop is always unstable regardless of what type of controller is employed. The implementation of the Smith predictor requires the known process characteristics which are obtained from the Process Reaction curve. The design of the controller then involves stability analysis using the Describing Function method.

The controller's settings depend upon both the specified dead zone of the switching relay and the allowed tolerance in the controlled output. When there is a change in either of them, retuning is required to ensure satisfactory performance of the controller. In addition, to maintain accuracy of measuring sensors recalibration should be done occasionally to minimize drifting effects on the output signal.

The computer program is developed for process control applications. In the real time, interrupt-driven environment, the program provides capabilities like data collection and storage for off-line analysis, system monitoring and control, operator-plant interaction. Extensive graphic display and keyboard assignment are utilized. The program can be enhanced to include on-line data printing; however this feature is not feasible because real time overhead is increased due to the added interrupt source.

REFERENCES

[1] Harry J.Sauer Jr. and Ronald H.Howell, '*Heat Pump Systems*', John Wiley & Sons 1983.

[2] F.Moser and H.Schnitzer,'*Heat Pumps in Industry*', Elsevier 1985.

[3] Kay H.Zimmerman, '*Heat Pumps: Prospects in Heat Pump Technology & Marketing* ', Lewis Publishers Inc. 1987.

[4] '*A Review of The Heat Pump Research and Development Program* ', Ontario Research Foundation 1986.

[5] Gordon J. Van Wylen and Richard E. Sonntag, '*Fundamentals of Classical Thermodynamics*', SI Version 2e, John Wiley & Sons 1978.

[6] K.Higuchi and M.Mayano, 'Dynamic Characteristics of TE Valves '. International Journal of Refrigeration Vol.5, No.4, July 1982.

[7] P. Danig, 'Liquid Feed Regulation by TEVs '. Journal of Refrigeration Vol.6-7, 1963.

[8] P.M.T. Broersen and M.F.G. Van der Jagt, 'Hunting of Evaporators Controlled by A TEV '. Transaction of ASME Vol.102, June 1980.

[9] W.D.Gruhle and R.Isermann, 'Modeling and Control of A Refrigerant Evaporator ', 1985 American Control Conference.

[10] W.D.Gruher, 'Digital Control of A Refrigerant Evaporator '. IFAC Digital Computer Applications to Process Control, Vienna 1985.

[11] Edward G.Pita, '*Refrigeration Principles and Systems: An Energy Approach* ', John Wiley & Sons 1984.

[12] D.R Wilson, R.K Green, D.F Neale, M.Searle, K.E Bird, S.A Tassou Y.T Wang, G. Nelson, B.Court and S.Harding, 'The Minimization of Power Consumption of A Thermodynamic Heat Pump By A Microprocessor-Based Control System', Commission of the European Communities. Report EUR7283 en MF, April 1981.

[13] Y.T.Wang, D.R.Wilson and D.F.Neale, 'Heat Pump Control ', IEE Proceedings Vol.130, No.6, November 1983.

[14] R.E.Rink, V.Gourishankar and M. Zaheeruddin, 'Optimal Control of Heat Pump/Heat-Storage System with Time-of-Day Energy Price Incentive ', Journal of Optimization Theory and Applications Vol.58, No.1, July 1988.

[15] '*Application Notes - Practical Temperature Measurements* ', Omega Engineering Inc.

[16] *'LabMaster DMA Handbook '*, Scientific Solutions Inc. 1987.

[17] Tung Chi Wah, 'An IBM PC and Tecmar-Based Digital Controller ', Master of Engineering thesis, University of Alberta, 1986.

[18] *'IBM Technical References '*, IBM Inc.

[19] Stephen Prata, *'Advanced C Primer ++'*, Howard W.Sams & Co. 1987.

[20] *'IBM Disk Operating System Technical References '*, IBM Inc.

[21] *'Intel 8088 Family User's Manual '*, Intel Corporation 1979.

[22] Ray Ducan, *'Advanced MS Dos Programming '*, Microsoft Press 1986.

[23] Goerge Stephanopoulous, *'Chemical Process Control: An Introduction to Theory and Practice '*, Prentice Hall Inc. 1987.

[24] C.Y. Chan and G.G. Haselden, 'Computer-Based Refrigerant Thermodynamic Properties. Part 1:Basic Equations', International Journal of Refrigeration Vol.4, 1981.

[25] C.Y. Chan and G.G. Haselden, 'Computer-Based Refrigerant Thermodynamic Properties. Part 2:Program Listings',International Journal of Refrigeration Vol.4, 1981.

[26] C.Y. Chan and G.G. Haselden, 'Computer-Based Refrigerant Thermodynamic Properties. Part 3: Use of the Program in The Computation of Standard Refrigerant Cycles ', International Journal of Refrigeration Vol.4, 1981.

[27] A.C. Cleland, ' Computer Subroutines for Rapid Evaluation of Refrigerant Thermodynamic Properties ', International Journal of Refrigeration Vol.9, November 1986.

[28] J.A. Miller, A.M. Lopez, C.L. Smith and P.W. Murrill, 'A Comparision of Controller', Control Engineering Vol.14, December 1967.

[29] J.G. Ziegler and N.B. Nichols, 'Optimum Settings For Automatic Controller', Transactions of ASME Vol.64, 1942.

[30] H.A. Fetik, ' Tuning Controllers For Noisy Processes', ISA Transactions Vol.14, No.4, 1975.

[31] D.P. Atherton, *'Nonlinear Control Engineering '*,Van Nostrand Reinhold Company, 1975.